WEB DESIGN PATTERNS FOR MOBILE DEVICES

Jorge Miguel Neves Ribeiro

MESTRADO EM MULTIMÉDIA

Faculdade de Engenharia da Universidade do Porto

Advisor: Miguel Carvalhais,
Faculty of Fine Arts, University of Porto

JULY 2012

ABSTRACT

We have in our hands a universe of personal devices that are always available, always on, and always connected. By their characteristics, mobile devices have brought new constraints and possibilities to the field of web design. It is no longer possible to design a website without at least consider how it will work on a mobile device: the whole mobile experience needs to be designed. Because designing for these devices is considerably different from designing for conventional web interfaces, we cannot only rely on the established techniques: we will need new ones.

Throughout a pattern-based approach, this work aims to contribute to the development of new tools for the design of mobile interfaces, by capturing and presenting already proven solutions that aspire to help us design better and more engaging experiences on the mobile web.

RESUMO

Temos à nossa disposição um universo de dispositivos móveis que estão sempre disponíveis, sempre ligados e sempre conectados. Devido às duas características, estes dispositivos trouxeram novas limitações e possibilidades ao domínio do web design. Não é possível desenvolver um website sem pelo menos considerar como é que irá funcionar num dispositivo móvel: toda a experiência móvel tem que ser pensada. E porque desenvolver para este tipo dispositivos é consideravelmente diferente de desenvolver para interfaces web convencionais, não podemos continuar a depender apenas das ferramentas atuais: precisaremos de novas.

Através de uma abordagem baseada em padrões de design, este projeto pretende contribuir para o desenvolvimentos de novas ferramentas que ajudem no design de interfaces móveis, através da captura e apresentação de soluções comprovadas que aspiram ajudar na criação de melhores e mais envolventes experiências web.

ACKNOWLEDGMENTS

I would like to thank my advisor Miguel Carvalhais for all the support and advice, essential to the unfolding of this work; to Dr. Alejandra Garrido for the help with the patterns; to the team at IDD who accompanied me through the past two year; to everyone that somehow contribute to the development of this dissertation; and to family and friends.

TABLE OF CONTENTS

INDEX OF FIGURES

1 INTRODUCTION

This work began from an interest to further explore the design of interfaces, particularly web interfaces, on mobile devices. We cannot remain indifferent to the growth of the mobile web, and to how it is changing the web as a whole, it is important and necessary to understand how it affects our work, and how we, designers, can contribute to the solution.

We have at our disposal a generation of portable devices that are closer to a personal computer than to a phone, that are very capable devices with more processing power than some computers from a few years ago, and with browsers that are comparable to their desktop counterparts. Devices are getting more affordable, and data connections are becoming cheaper, faster and ubiquitous. By their portability, mobile devices accompany us to wherever we go and are used on the most diverse contexts, for the most distinct purposes, what Clark (2010) describes as "nontraditional computing environments". The web usage has also been changing, more people are using their devices as an effective mean to access the web, and for some users that is indeed the only, or at least the primary, channel to browse the web.

We have a new interaction model, based on gestural interfaces that provide for more natural and appealing interactions. But because of its relative novelty, there are no strong standards or guidelines to help us design for them, there is in fact a lack of established guidelines for gestural interfaces (Nielsen and Norman 2010). Moreover, there is a lack of recognized patterns. Users do not have rigid expectations of how an interface should behave (Budiu and Nielsen 2010, 6), as they might have in desktop environments. The usage of gestural interfaces became more exploratory, i.e., when lost, or just because they want to explore the interface, they tap arbitrarily just to see how interface elements respond. Nevertheless, we can regard this not necessarily as problem, but as an opportunity to experiment with new approaches to the design of sites.

These devices shifted, in an irreversible manner, the perception of what is the web, and have raised new questions of how to design for it. Designing for mobile should be an essential part of any present design project, in fact, it should be considered from the beginning of every project. The mobile and desktop are not incompatible or even opposing forces, they are in-

deed part of the same web experience that needs to be designed as a whole. We can now start thinking about creating unified web experiences between mobile and desktop environments.

Designing for mobile need to take into account the mobile landscape, the characteristics of these devices, and the limitations and possibilities that they impose in order to understand what their role on the web is. Designing for mobile devices is considerably different from designing for traditional desktops environments, so the tools that we have now may not be enough for our needs. It is important to rethink our current tools and strategies in order to understand how they can be used in our advantage.

Like any new medium, mobile devices impose new constraints and capabilities that should be considered, but these limitations are what make the mobile web a much more challenging and interesting subject within the field of web design. There is a more receptive space to the conception of new ideas of what can be considered an interface; there is a gap to explore, develop and suggest new techniques, or more precisely, to find and propose new patterns.

1.1 Objectives

In this work we intend to explore the design of web interfaces on mobile contexts, which invokes the need to study the implications that these devices brought to the field of web design. It is essential to outline and understand what are its main characteristics and limitations, and how they affect the design of sites, especially when compared to traditional web interfaces. Therefore, in this research, we expect to summarize the prevailing factors that one needs to consider when designing a mobile website.

We intend to explore the design of mobile interfaces through a pattern-based methodology. In order to accomplish that objective, it is important to study what are design patterns and how they can be used as a design tool. We therefore aim to identify, understand and systematize different methodologies that can be applied to the writing, cataloguing and presentation of design patterns, with the purpose to propose a template that best fit the patterns in this work.

The main objective of this dissertation is to develop a set of design patterns that capture proved solutions, and describe them in a comprehensible manner so they can be easily used by people throughout their design projects. We intend to present those patterns in an organized catalogue and offer them through an online platform accessible to the design community.

Therefore, with this project we expect to contribute to the development of a pattern library dedicated to the design of web interfaces for mobile devices, with which we hope to endow designers and non-designers with a set of useful solutions and tools that augment their capabilities to design and evaluate websites. Ultimately, the goal of this project is to indirectly improve the end-user fruition of their devices when browsing the web.

1.2 Dissertation Structure

The first chapter introduces the context of this dissertation, and describes the motivation and objectives that will lead us throughout the development of this project.

The second chapter addresses the concerns regarding the design of sites for mobile devices. It defines our object of study and enumerates the constraints and possibilities that these devices impose.

The third chapter introduces the concept of patterns and pattern languages, and provides a theoretical background for the work in the following chapters.

The fourth chapter presents a comparative analysis of pattern libraries, proving a historical background. It presents an individual study of each library, as well as a critical analysis of the different approaches to the problem of presenting patterns.

Based on the research conducted on the previous chapters, the fifth chapter describes the template that will be used in the patterns of this dissertation.

The sixth chapter unveils the twenty-one design patterns that comprise the main body of this dissertation, a library of patterns that synthesizes the research performed in this work. The seventh chapter presents three cases studies, each one describing the process of implementing the patterns that were previously formulated on a real web design project.

Finally, the eighth chapter summarizes the results of this research, reporting the contributions and future work.

2 DESIGNING FOR MOBILE DEVICES

2.1 The Devices

In order to understand how to design for mobile, first we need to define what a mobile device is. Hoober and Berkman (2011, xvi) consider that a mobile device has the following characteristics: it is small, portable, connected, interactive, and contextually aware. Firtman (2010, 4) describes a mobile device as being: portable, personal, available all the time, easy and fast to use, and having some kind of network connection.

These definitions are too broad in such a manner that may describe as *mobile* an enormous set of devices. It is possible to include in the description everything ranging from smartphones, feature phones, portable gaming devices, portable media players or tablets. To address this problem it may be helpful to categorize them. Firtman (2010, 6–10) groups mobile devices in the following categories: mobile phones, low-end mobile devices, mid-end mobile devices, high-end mobile devices, smartphones, non-phones devices, small personal object technology and at last, tablets, netbooks, and notebooks. Devices from the first group do not have Internet access, so we can leave them aside; tablets and netbooks, although portable, may deserve an entire category for themselves; and what the author describes as low-end, mid-end, high-end mobile devices are devices that fit into a category that is commonly designated as *feature phones*. Alternatively, Fling (2009, 3–10) sets a chronological background for the evolution of mobile devices — he only examines mobile phones though. He places the 'feature phones era' between 1989 and 2008, the 'smartphones era' from 2002 to present and at last the 'touch era', which began with the release of the iPhone in 2007. Chronological categorization may not be always helpful, since, for example, a smartphone can also be a touch device.

On the other hand, Wroblewski (2011) only distinguishes between feature phones and smartphones, though he never explicitly defines them. Likewise, Hoober and Berkman (2011) only compare feature phones to smartphones, but do not describe them. Koch (2010) only refers to smartphones and 'non-smartphones', and proposes a simpler definition: "A smartphone is a phone that runs a recognizable OS on which the user can install applications." However, he

divides smartphones in three categories — analogous to those used by Fling (2009) — based on their audience and cost: mid-range, business, and high-end.

It is possible to recognize that the line that distinguishes a smartphone from a feature phone is not that clear or even consensual, and to make it worse, the definition of what a smartphone is constantly evolving (Firtman 2010, 8) and changes depending on the author. If we understand the characteristics that define what a smartphone is as transitional, the term *smartphone* can be used to describe the most advanced device at a given time. In general, feature phones have more capabilities than a simple cell phone but not enough to be called a smartphone; they have a proprietary operating system (usually firmware), which cannot be easily updated; and the support for third-party applications is limited or nonexistent. Although these devices are not web oriented they have some sort of web support and offer a basic browser. Smartphones have all the functionalities of a feature phone as well as a multitasking operating system, a full desktop browser, Wifi, 3G or 4G connections, a camera and an additional set of sensors (e.g., accelerometer, GPS) (Firtman 2010, 8), and can install third-party applications.

2.1.1 Operating Systems

Linked to the device we have the underlying operating system. When users decide to buy a phone, they are also choosing the operating system they will use, which helps to distinguish if the device is a feature phone or a smartphone. The operating system, but also its version, is important given that it dictates what will be the browsers available for that device.

According to *Statcounter,*[1] on the first trimester of 2012 the worldwide leading operating system on mobile devices was Symbian OS with 31.22%, immediately followed by iOS with 24.47%[2] and Android with 23.83%. However, if we only take in account the data from Europe or the USA these numbers change dramatically. Symbian drops to 8.37%[3] in Europe and to 1.66%[4] in the USA; iOS jumps to the first place with 41.69% in Europe and 47.02% in the USA; Android reaches the second place with 31.33% in Europe and with 41.42% in the USA. Both are followed by Blackberry OS in the third place but with a much lower market share.

Another relevant factor is the ratio of devices that use the latest version of the OS, because a more recent version generally means a better browser and support to new features. On Android only 7.1%[5] of the devices have the latest version installed. Most devices (64.6%) are with

---

1 Source: http://gs.statcounter.com/#mobile_os-ww-monthly-201201-201203-bar.

2 This value represents all iOS devices; it does not distinguish whether it comes from the iPhone, iPod or iPad.

3 Source: http://gs.statcounter.com/#mobile_os-eu-monthly-201201-201203-bar.

4 Source: http://gs.statcounter.com/#mobile_os-US-monthly-201201-201203-bar.

5 According to the official statistics from Google (http://developer.android.com/resources/dashboard/platform-versions.html), which is based on the Android devices that accessed Google Play during a 14-day period ending on June 1, 2012. The value 7.4% is the result from combining the minor versions of Android 4.

version 2.3 and a respectable amount (19.1%) are still using version 2.2 that was released in May 2010. On the iOS platform the adoption rate of the new versions is noticeably faster: around 72%[6] of iOS users have already updated their devices to the latest version.

2.1.2 Performance

Mobile devices are also more limited in terms of performance than a common desktop or laptop computer. This does not make a very huge impact on simple pages, but is easier to notice, for example, on pages with complex animations on slower devices. It is important to be aware how the technical implementation affects the user experience, because simple CSS properties, such as *box-shadow*, or even how the CSS itself is coded, can make a negative impact on the page performance.

Likewise, the network performance should not be disregarded. Mobile devices can be used in outdoor environments, where the only Internet access available is through mobile networks that are slower, and have generally expensive data plans. Serving oversized images can make a large impact on the page loading time, and thus, affect the overall experience of the site. Mobile websites — and websites in general — need to be optimized so that they do not receive unnecessary assets. Some new CSS properties (e.g., *box-shadow*, *linear-gradient, font-face*) allows us to stop sending images that are usually heavier, while still having rich graphics enhancing the page; however, one can use, and should use, any technique that reduces and optimizes the content that is delivered to the device.

2.1.3 Why only smartphones?

Although we recognize that a website should be accessible by any kind of device, and despite feature phones representing a significant percentage of mobile devices, we think that it is more worthwhile to focus our efforts — as designers in general, and in particular for the purpose of this dissertation — on designing for smartphones. Wroblewski (2011) points out a few reasons behind his decision to focus on designing for smartphones: smartphones have a disproportionate amount of web and data usage[7] — smartphones represent only 13% of total mobile device but account for over 78% of total mobile traffic; smartphones have a faster adoption rate[8] — in the fourth quarter of 2011 smartphone sales grew up 47%; and they are getting more and more

---

6 Apple does not make statistics publicly available. This value comes from the data of one developer — David Smith (http://david-smith.org/blog/2012/05/11/ios-5-dot-1-1-upgrade-stats) — that has a data sample of 100 000 weekly downloads.

7 Source: http://newsroom.cisco.com/dlls/ekits/Cisco_VNI_Global_Mobile_Data_Traffic_Forecast_2010_2015.pdf.

8 Source: http://gartner.com/it/page.jsp?id=1924314.

affordable. This idea it is also supported by Koch (2010) because, as he puts it, "all good mobile browsers run on one smartphone platform or another".

Moreover, Nielsen recently dropped user testing with features phones (2011) and enumerated three reasons for that decision:

- "[...] feature phone usability is so miserable when accessing the Web that we recommend that most companies don't bother supporting feature phone.
- Empirically, websites see very little traffic from feature phones [...];
- Pragmatically, almost all participants in our mobile user experience courses tell us that they don't design for feature phones."

Even though designing for only a subset of devices might pose the risk of alienating a respectable share of users, the adoption rate and the greater experience that these devices offer, make it more compelling to do so.

2.2 The Browsers

Since we are talking about designing for the web, it is important to understand which browsers are available, what are their capabilities, and how they compare to desktop browsers. The browser used, and its capabilities will dictate how the page will render, and thus, contribute to how the website will be experienced.

According to *StatCounter* the worldwide mobile browser share[9] on the first quarter of 2012 was led by Opera with 22.36%, followed by the Android browser with 21.6% and Mobile Safari with 20.16%. However, if we take as reference Europe and the USA those numbers change considerably. In both cases Opera drops abruptly, to 1.98%[10] in the USA and 10.14%[11] in Europe. On the other hand, in Europe the iPhone gets 36.09% followed by the Android with 29.15%; in the USA, Android leads with 40.43% followed with the iPhone with 37.8%. Except for the Blackberry and Nokia browsers, the market share of all other browsers is almost residual.

On the desktop we have five major browsers (Internet Explorer, Firefox, Chrome, Safari and Opera), which represent four major layout engines (Trident, Gecko, Webkit and Presto); on mobile, the browser engines are not much different, but the leading browsers are others though. When we take Europe or the USA as reference, more than 70%[12] of current mobile web browsing comes from Webkit browsers (essentially Safari Mobile and the Android browser), and because Webkit represents a great percentage of mobile browsers we can expect similar

---

9 Source: http://gs.statcounter.com/#mobile_browser-ww-monthly-201201-201203.

10 Source: http://gs.statcounter.com/#mobile_browser-us-monthly-201201-201203.

11 Source: http://gs.statcounter.com/#mobile_browser-eu-monthly-201201-201203.

12 When the results from the Safari Mobile and Android browser are combined.

web rendering on simple web pages (Firtman 2010, 44); however, there are differences between Webkit implementations, so testing on only one device and browser does not guarantee that a page works perfectly on every Webkit browser.

Moreover, mobile browsers are in general more limited in terms of features than their desktop counterparts, but more advanced than older[13] desktop browsers. For example, CSS gradients are supported by most mobile browsers but are still unsupported by Internet Explorer. Some features are not available, or have limited support on mobile. For example, the support for fixed positioned elements is incomplete, or even nonexistent on almost all the current mobile browsers; and the support for SVG files is absent from older Android versions. The supported features depend greatly on the browser and devices used, so ideally websites should be tested in as many devices and browsers as possible.

Fling (2009, 170) proposes a classification of mobile browsers based upon their characteristics, from Class A to Class F. Class A mobile browsers have an excellent support for HTML, CSS and JavaScript that is comparable to a "desktop-grade" browser. On the contrary, Class F are browsers with unreliable CSS support, poor table support, basic forms, and no JavaScript support. Lower-grade browsers are usually associated to feature phones and older devices, whereas Class A browsers are present on modern devices such as the iPhone and Android devices. Class A browsers offer a superior browsing experience, and for the same reasons that were stated above for smartphones, they are the ones that you will want, and probably should, focus your work.

There is also an additional class of browser, in which the browser client does not contact the website server directly. Instead, all content goes through a proxy server that performs a series of actions on behalf of the client and reformats the page to a format more suitable for the client device. Browsers such as the Opera Mini or Skyfire are included in this class of browsers, and can perform the following actions (Firtman 2010, 43): reduce the content by eliminating features that are not compatible with the device; compress the content; pre-render the content so it can be displayed in the browser faster; convert the content to a supported format, e.g., Flash; encrypt the content; and cache the content for quick access of frequently visited sites. These browsers are very limited in terms of dynamic interactions, since that for every new state the browser request information from the server, so if you need to support them, more complex interactions need to be thought with caution.

---

13 We are referring to browsers that were released years ago but still have a respectable user share, and need to be somehow supported, e.g. Internet Explorer 7.

2.2.1 Browser Chrome

> *"Chrome" is the user interface overhead that surrounds user data and web page content. Although chrome obesity can eat half of the available pixels, a reasonable amount enhances usability.* (Nielsen 2012a)

The user interface that surrounds the web page is part of the mobile experience that should be taken into consideration. While it is not that significant on desktop environments, the interface of the OS and browser takes up considerable space on smaller devices, which is a scarce asset on mobile.

On the first level there is the operating system interface. Usually it is presented as a status bar on top of the screen that provides access to system-wide information and functions. On a second level we have the browser interface, which varies depending on the operating system and browser. The space assigned to the page content differs, as well as the functionalities offered. For instance, iOS devices have a navigation bar at bottom that allows users to go back and forward between pages, while Android devices only have a hardware button to go back and thus dispense the chrome (Figure 2.1).

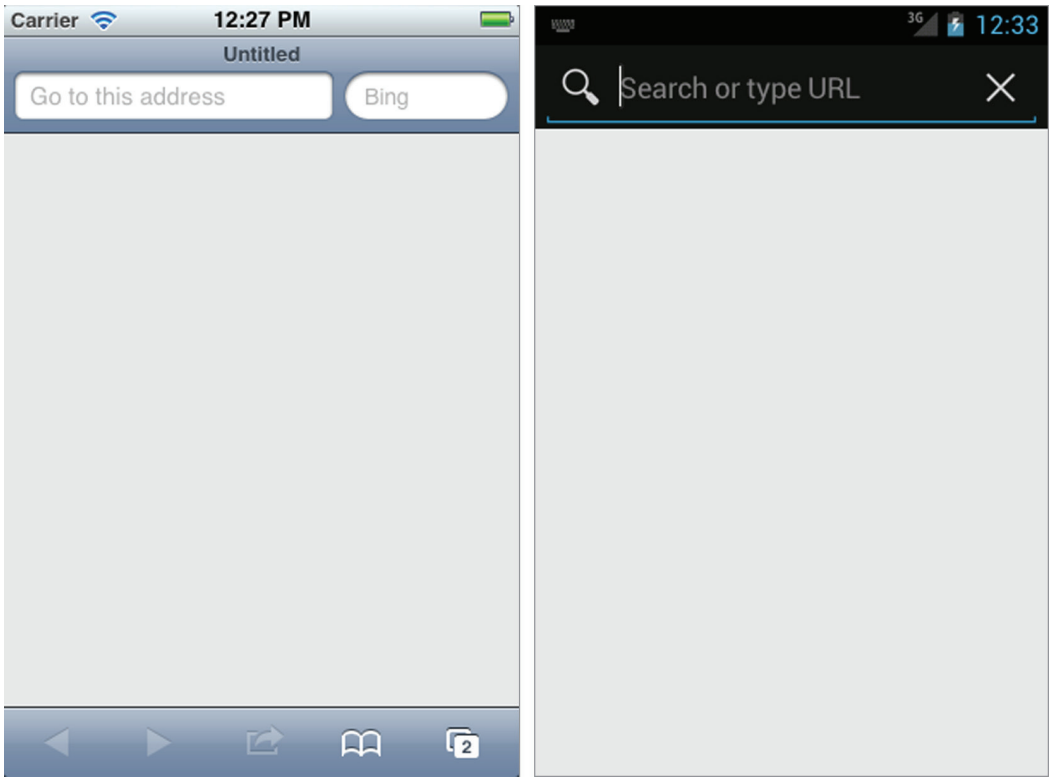


Figure 2.1 Default browser chrome on iOS (left) and Android (right).

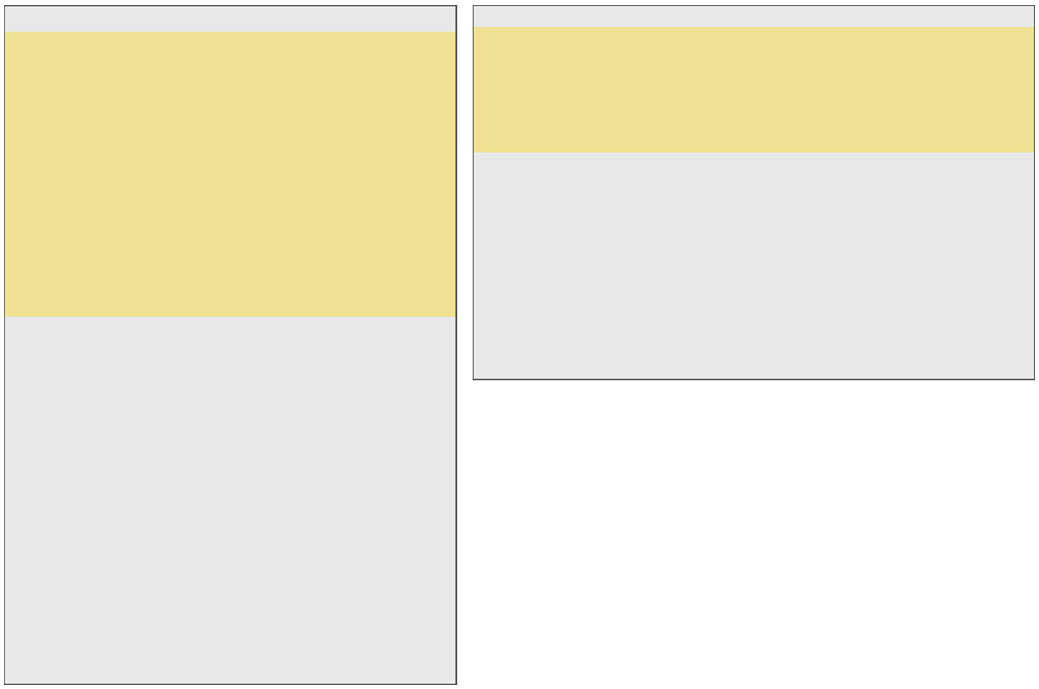


Figure 2.2 Available area (marked in yellow) when using a virtual keyboard in iOS; portrait on the left, landscape on the right.

In addition to the permanent chrome we should not forget the interface elements that appear on specific moments. For example, the virtual keyboard that appears of text forms can make up to more than half[14] of the screen (Figure 2.2).

2.3 Screen Sizes and Resolutions

A noticeable difference between desktop and mobile environments, and probably the most significant limitation that one faces when is designing or adapting a website for a mobile view, is the screen size: mobile devices are considerably smaller than their mobile counterparts. Users are accustomed to browse the web on desktop and laptops that have screens with more than thirteen inches of diagonal, whereas mobile devices screens are around three inches.

On the desktop, 98%[15] of users browse the web on displays with screen resolutions equal or higher than 1024x768 pixels, while on mobile these values are, as it is expected, much lower and vary significantly with the type of device. According to *Statcounter*, on the first trimester of

14 If we took the iPhone as reference, 58% on portrait, and 67% on landscape view is dedicated to OS and browser chrome.
15 Based on the statistics from W3Schools (2012) from January 2012. It reports only from accesses on their website, so it might not be representative of all websites.

2012 the most common screen resolution on mobile devices was 320x480 pixels with 20.43%.[16] When the 1024x768 pixel resolution is taken as reference and compared to the most common resolution of mobile, we lose around 80% of screen space (Figure 2.3). With less available space, websites need to be rethought and stripped down to its essence (Tidwell 2011, 442).

Figure 2.3 Desktop (1024x748) vs mobile (320x480) screen sizes.

Another related problem is the variability in screen size and the different proportions that are available in the current mobile device landscape. Screen heights are not so problematic because most websites structure the page vertically. The 320x480 pixel resolution is the most common, but we are faced with a myriad of devices with a diverse range of screen sizes and resolutions. *OpenSignalMaps* clearly illustrated the problem of screen fragmentation of Android devices on Figure 2.4.

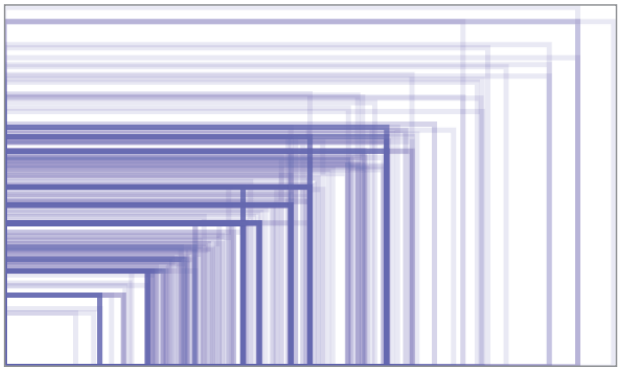


Figure 2.4 Different screen sizes on Android devices. Data collected from 681 900 devices on 195 countries. The color opacity represents the incidence of that resolution. Source: *OpenSignalMaps*.[17]

16 Source: http://gs.statcounter.com/#mobile_resolution-ww-monthly-201101-201205-bar.

17 Source: http://opensignalmaps.com/reports/fragmentation.php.

We only find a more stable scenario on the iOS platform. The iPhone only has two different resolutions. Older devices have a resolution of 480x320 pixels and the new ones have a resolution of 960x640, with the same screen size but with the double pixel density. However, sizes set in pixels do not correspond to physical pixels on high-resolution devices. Because a 960x640 screen has twice the resolution of a 480x320 screen, pixels on higher-resolution screens are mapped to a virtual 2x2 pixel square, which has the same physical size of a pixel on lower resolution devices. At first glance, the same website when viewed on devices with different screen densities will look similar, with the exception of the bitmap graphics that can appear blurred, especially when compared to the typography that scales according to the screen resolution. This compels us to reflect on the need of delivering resolution independent assets, e.g., vector images, which, like typography can be scaled to different sizes without losing quality.

The proliferation of devices with distinct sizes and resolutions impose the need to design interfaces that are not bound to a particular width or height. We can no longer foresee which devices will access our websites, and designing a different layout for each device is not a very practical, or even feasible, solution. According to Nielsen (2012b), if you can afford it, you should build a separated mobile-optimized website because they offer a better experience; however, a design strategy based on the principles of a *responsive design* (Marcotte 2010; 2011), in which we design a fluid layout that adapts gracefully to different screen sizes, is a much more pragmatic approach to the problem of designing websites that need to work on the plenitude of current and future mobile devices, as well on desktop environments.

2.4 Touch Interfaces

> [Physical gestures] *can enhance the pleasure and engagement of participants. They can even be used as exercise machines. But they also can do damage.* (Norman 2010)

A significant difference between desktop and mobile interfaces is the way in which we interact with them. Traditional desktop environments rely primarily on mouse-driven interactions, whilst mobile interfaces are touch-oriented. As it was stated earlier, although feature phones with qwerty keyboards and keypads still account for a considerable percentage of the current devices, the trend is towards the growth of touch and multitouch devices.

Unlike conventional mouse-driven interfaces, in which there is an artificial pointing device that acts as an intermediary between our movements and what happens on the screen, e.g., a mouse or a trackpad that controls a cursor on the screen, touch interfaces allow users to interact directly with the objects on the screen. Gestural interfaces become " [...] the ultimate in direct manipulation: using the body to control the digital [...] space around us" (Saffer 2008, 4). On touch interfaces the content becomes the UI, and the UI becomes the content.

There are also human constraints that drive the design of mobile interfaces. The human finger is a much more imprecise pointer than a mouse, so it is not easy to accurately hit targets that are much smaller than the size of the fingertip (Budiu and Nielsen 2010). With a mouse, and with some effort, it is possible to hit a target as small as one pixel, task that is almost impossible to achieve without some trial and error on a touch screen. This may inflict even bigger problems for people with less dexterity, particularly older people, and for the "rushed and distracted user" (Clark 2010, 13). Therefore, targets on touch interfaces need to be properly sized and positioned (Wroblewski 2011, 67) in order to account for the limitations of the user and the technology. Touch interfaces also have their advantages, besides the more natural and pleasing interactions, it is possible to explore the use of gestures, — e.g., the pinch to zoom —, as a way to improve the user experience.

This is further worsened because current touch screens do not offer any haptic feedback.[18] On hardware keyboards we get a physical response when a button is pushed; however, touch screens are solid flat surfaces that do not move when pressed. The only feedback that users receive is the one that is provided by the interface itself, whether it is visual or audible.

Touch interfaces are also more propitious to accidental selection and triggering of actions (Nielsen and Norman 2010). This problem occurs when targets are small and placed too close together so that users may hit the wrong target; when users rest some finger on top of the screen; or when they are holding the device and some part of their hands touch the screen by mistake. Moreover, because touch screens are used with the users' hands over the screen, the hand may block the view of an important part of the interface.

With regard to the browser, another major difference is the lack of the hover state. The mouse-over technique is a common pattern on the web desgin that is used for revealing additional information, e.g., tooltips, but it does not work on touch devices because there is not a cursor to hover on page elements. Therefore, interactions that rely merely on mouse-over need to be rethought (Wroblewski 2011, 78), so no critical information is left behind hover-based interactions.

In spite of the limitations, touch interfaces offer more natural and pleasing experiences that are favored to those obtained by keyboards in mobile devices. But most of all, these restrictions should to be understood as a way to explore and propose fresh approaches to the design of interfaces.

---

18 We can use the phone vibration as a way to emulate a haptic feedback. However, currently, only Mobile Firefox has some experimental support for the Vibration API.

3 DESIGN PATTERNS

Having found and described some concerns with the access and visualization of web pages in mobile devices, we decided to address the problem of designing for these devices through a pattern-based methodology. Foremost, it was necessary to understand what is a design pattern and how they can be a valuable artifact in the design process.

3.1 What is a Pattern?

> *Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice.* (Alexander 1977, x)

A pattern consists of a description of a recurring problem, and the solution to that same problem within a context. They are means to capture abstract and well-proven solutions for concrete real-world problems, and to document that knowledge in a way that can be shared with others. Borchers defines patterns as a "structured textual and graphical description of a proven solution to a recurring design problem" (2000, 7). For van Duyne (2006, 19) they "communicate insights into design problems, capturing the essence of the problems and their solutions in a compact form". Tidwell (2011, xviii) argues that patterns capture a common structure without being too concrete on the details, which gives designers the flexibility to be creative.

The use of patterns as a design tool was introduced in 1977 through the work of the architect Christopher Alexander and his team on the study of design patterns applied to architecture and urbanism, with the publishing of *A Pattern Language* (Alexander et al. 1977), thought the theoretical background was set earlier on *Notes on the Synthesis of Form* (Alexander 1964) and *The Timeless Way of Building* (Alexander 1979). Nevertheless, the concept of patterns did not get much traction with his fellow architects, but was successfully adopted by software engineers and incorporated on the principles of object-oriented programming, which achieved popularity in the 1990s with the publishing of the book *Design Patterns: Elements of Reusable*

Object-Oriented Software (Gamma et al. 1995). Afterwards appeared the first attempts to adapt the work of Alexander to the field of HCI. A first substantial set of patterns was proposed by Jenifer Tidwell in *Common Ground* (1999), and later expanded on works such as *The Design of Sites* (van Duyne, Landay and Hong 2006) or *Designing Interfaces* (Tidwell 2005), as well as in others that will be discussed later. Despite the shift on the domain of application of patterns, the essential characteristics that define a pattern remained.

A pattern is a three-part rule that expresses a relation between a certain context, a problem, and a solution (Alexander 1979, 247): a *context* that sets up the pattern in the language; a *problem* that describes a common problem; and the *solution* that prescribes a response to that same problem. Formally the structure of a pattern differs depending on the domain — this will be explored further on the next chapters — but nonetheless, each pattern contains only the essentials that cannot be avoided if one wants to solve the problem (Alexander et al. 1977, XIV), so nothing should be imposed.

Patterns are possible solutions to a specific design problem, but nevertheless, they are only suggestions, or as Alexander puts it "each pattern represents our current best guess as to what arrangement of the physical environment will work to solve the problem present" (1977, XV).

The goal of a pattern is to solve a problem by the means of resolving or balancing the 'forces' that are conflicting in a given context. For example, an *Entrance Room* should be designed in such a way that addresses the following: "A person answering the door often tries to see who is at the door before they open it." (Alexander et al. 1977, 623). Forces are the constraints that drive a design, and that should be answered before a system finds its equilibrium. The nature of forces depends on the domain of the pattern but can be of social, economic, or physical nature (Borchers 2001, 11).

A pattern is an attempt to discover some invariant features that occur on a particular system of forces (Alexander 1979, 260). That is not always successfully done. Patterns are not all equal and authors cannot be equally confident about the validity of all patterns. Therefore, it is important to assess the authors' confidence on any pattern. Alexander defined that confidence in terms of the 'invariant'. That is, there are unavoidable features that are present on all the successful implementations of a given pattern, so that a solution for a pattern will become more 'invariant' the more it captures those features. For instance, Alexander asserts that a pattern is a 'true invariant' when the authors are very confident that it is not possible to solve that problem without using the prescribed solution.

3.1.1 Interaction Design Patterns

As it was stated above, the concept of patterns was later embraced in domains other than the architecture and urbanism. Tidwell's (1999) and Borchers's (2001) works introduced the concept of patterns to the domain of interaction design, which, while different, share some similarities

with architecture: they both produce artifacts that inhabitants (or users) can directly interact with or even live in (Borchers 2000).

A key distinction between architecture patterns and interaction design patterns is the inclusion of the time dimension. Alexander's patterns deal with the spatial relations between static structures, which is also reflected on how the patterns are organized in the language: from towns to the smaller construction details. Interaction design patterns, in addition to the spatial dimensions, also have to cope with the dynamic behaviors and change in elements in response to the user activity (Cooper, Reimann and Cronin 2007, 156); changes that occur both in response to the application state and the human activity. Interaction design patterns are more dynamic, and the context and forces often change during the course of interaction (Borchers 2001, 28).

This distinction is particularly important when one needs to illustrate a pattern. For example, Alexander's patterns are illustrated by photos of towns and buildings, however, the dynamic components of interactive design patterns cannot be easily captured by this type of illustration. Therefore new strategies need to be found.

3.2 Pattern Languages and Pattern Libraries

> [...] *when you build a thing you cannot merely build that thing in isolation, but must also repair the world around it, and within it, so that the larger world at that one place becomes more coherent, and more whole; and the thing which you make takes its place in the web of nature, as you make it.* (Alexander 1977)

Patterns are not isolated entities. To be useful they must be meaningfully organized on a set of patterns related to a specific context (Cooper, Reimann and Cronin 2007, 157). Patterns exist on a structured network of interdependent patterns; each one depending both on the smaller patterns it contains, and on the larger patterns within which is contained (Alexander 1979, 312). This means that patterns are linked in a system, in which each one depends on all the others, so to solve a design problem all nodes in that system must be resolved.

A set of patterns forms a language, and any smaller sequence of patterns of that language also forms a language on its own. Alexander uses the term *pattern language* to describe all patterns in a set (e.g., all 253 patterns in *A Pattern Language*), as well as to describe any smaller sequence of patterns. Borchers defines a pattern language as a "hierarchy of design patterns ordered by their scope" (2001, 7), while Tidwell (2011, xix) defines it just as a very complete set of patterns. A more common term to describe a collection of patterns in a domain is *pattern library* or *pattern catalogue* (Cooper, Reimann and Cronin 2007, 157); whereas, they define a *pattern language* as a set of patterns that "is rigorously defined and specified, and sufficiently

complete to describe all solutions in a domain". The exact definition of what is a *pattern language* and its precise scope may vary depending on the author; nevertheless, in short, a *pattern language* is a collection of patterns for a given context.

In this work we adopted the term *pattern library* to describe a collection of related patterns for a specific domain, e.g., the collection of all the patterns in this work or in books as *A Pattern Language* (Alexander et al. 1977); and *pattern language* to describe any subset of patterns collected from a pattern library that are used on a specific context, e.g., all patterns that form the interface of a particular website.

3.3 The Benefits of Design Patterns

The goal of Alexander's patterns was to empower inhabitants with the tools to design and create their own buildings. Patterns support this goal by capturing well-fitting designs and writing them down in an accessible and comprehensible language, understandable by the laymen. They capture the collective wisdom of designers in a way that can be use used by less-experienced designers (Tidwell 1999), by providing designers with a set of reusable design solutions that can be used on multiple projects and shared with others. They also offer some advantages against other design artifacts, such as design guidelines, style guides or standards. Design guidelines are too abstract or too concrete. Abstract guidelines do not provide a solution to a real problem and do not create a common vocabulary of applicable solution, therefore they do not form a language (Borchers 2001, 7). On the other hand, concrete guidelines only work for a specific interface, which makes them quickly obsolete. Style guides are to used ensure that interfaces share a coherent look and feel; however they are too much tied to a specific technology or company, so they cannot be used outside that particular domain; whereas, patterns do not make assumptions about the aesthetics of a design. Standards establish rules and technical specifications that oversee the implementation of a particular technology on a given domain (e.g., the W3C Web Standards); however, standards are too abstract, do not provide a context, lack examples of use, and are usually written on a very formal and almost unintelligible language that cannot be easily comprehended by people outside that particular technical background. Patterns have had great acceptance on the HCI community, because they exhibit some benefits:

- Patterns provide designers and non-designers with a common vocabulary and terminology that is valuable in the communication within a team, but also to communicate a project to stakeholders (Tidwell 1999). All the involved parties in a discussion can talk about the same concepts by referring to the patterns name, which results in fewer misunderstandings.

- Patterns can work as a learning and educational tool (Carvalhais 2008; Tidwell 2011). They allow designers to expand their interface design vocabulary by reading patterns as a source of inspiration or as reference for when a problem arises.
- Patterns reduce design time and effort on new projects (Cooper, Reimann and Cronin 2007, 156), because they offer designers a set of reusable and already proven solutions, thus suppressing the need to solve problems that were already solved;
- Moreover, because patterns solve common problems, they allow designers to devote more time to solving new problems. They impose constraints to the designer's work but allow flexibility in the visual design and interaction details (Tidwell 1999).
- Patterns reduce the users' cognitive load by serving solutions that are familiar. The widespread use of a given pattern makes the behavior of that interface component more easily recognizable, and thus, reduces the time that one needs to learn a new interface.

4 COMPARATIVE ANALYSIS OF PATTERN LIBRARIES

4.1 Pattern Libraries

In order to better understand the writing of patterns, and to help us to find a template for the patterns in this work, we started by developing a comparative analysis of sixteen libraries:

- *A Pattern Language* (Alexander et al. 1977);
- *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma et al. 1995);
- *Common Ground: A Pattern Language for Human-Computer Interface Design* (Tidwell 1999);
- *A Pattern Approach to Interaction Design* (Borchers 2001);
- *The Design of Sites: Patterns for Creating Winning Websites* (van Duyne, Landay and Hong 2006);
- *Designing Interfaces* (Tidwell 2011);
- *Yahoo! Design Pattern Library* (Yahoo! 2006);
- *Patterns for Computer-Mediated Interaction* (Schümmer and Lukosch 2007);
- *UI Patterns* (Toxboe 2007);
- *Info Design Patterns* (Behrens 2008b);
- *Patternry* (Pattern Factory 2009);
- *Designing Web Interfaces* (Scott and Neil 2009);
- *Designing Social Interfaces* (Crumlish and Malone 2009);
- *Banco de Padrões de Design* (Instituto Superior Técnico 2010);
- *Designing Mobile Interfaces* (Hoober and Berkman 2011);
- *Mobile Design Pattern Gallery* (Neil 2012).

The main focus of the analysis was on libraries with patterns for interaction design, especially those with emphasis on the design of sites. We sought to include in this study a diverse range of libraries, regarding how they addressed the organization of patterns; the list should also reflect what we considered the most relevant libraries for the domain of this work. Nonetheless, two exceptions were made for historical reasons — *A Pattern Language* (Alexander et al. 1977), which introduced the concept of patterns, and *Design Patterns: Elements of Reusable*

Object-Oriented Software (Gamma et al. 1995) that proposed the first meaningful set of patterns outside the domain of architecture and urbanism. Two of these libraries contain patterns for the design of mobile interfaces: *Designing Mobile Interfaces* (Hoober and Berkman 2011) and *Mobile Design Pattern Gallery* (Neil 2012), although the last one does not contain patterns as they are usually understood — it is mainly a gallery of images. With regard to their medium, seven of the libraries are books, five are online libraries and the remaining are available on both media.[19]

The analysis was conducted by taking the model defined by Alexander as a starting point and comparing it to those used in other libraries. We focused on two areas: how patterns are organized in the language; and how the pattern itself is composed.

In the analysis of the organization of patterns in the language we focused on:

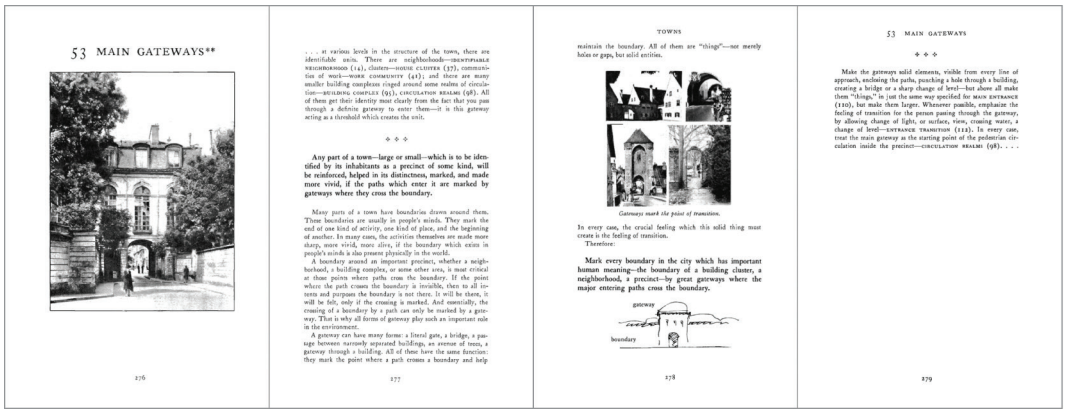
- what hierarchy was used;
- the characteristics of each hierarchical level (when they exist);
- how patterns were ordered.

For the analysis of the structure of the pattern itself we took the sections used by Alexander — name, ranking, picture, content, problem, forces, solution, diagram, and resulting context — and compared them with those used in other libraries. We focused on:

- the layout used to format the pattern;
- the writing style;
- which sections were used, and which were left out;
- how authors explained each section;
- the different nomenclatures given to the sections;
- other sections not present in Alexander's work that seemed relevant.

19 We only considered libraries in which the online and print versions were the same or very similar. For example, *Designing Social Interfaces* (Crumlish 2009) has a website where the names of the patterns are listed, but it is mostly a promotional website for the book.

4.1.1 *A Pattern Language*


53 MAIN GATEWAYS**

. . . at various levels in the structure of the town, there are identifiable units. There are neighborhoods—identifiable NEIGHBORHOOD (14), clusters—HOUSE CLUSTER (37), communities of work—WORK COMMUNITY (42); and there are many smaller building complexes ringed around some realms of circulation—BUILDING COMPLEX (95), CIRCULATION REALMS (98). All of them get their identity most clearly from the fact that you pass through a definite gateway to enter them—it is this gateway acting as a threshold which creates the unit.

Any part of a town—large or small—which is to be identified by its inhabitants as a precinct of some kind, will be reinforced, helped in its distinctness, marked, and made more vivid, if the paths which enter it are marked by gateways where they cross the boundary.

Many parts of a town have boundaries drawn around them. These boundaries are usually in people's minds. They mark the end of one kind of activity, one kind of place, and the beginning of another. In many cases, the activities themselves are made more sharp, more vivid, more alive, if the boundary which exists in people's minds is also present physically in the world.

A boundary around an important precinct, whether a neighborhood, a building complex, or some other area, is most critical at those points where paths cross the boundary. If the point where the path crosses the boundary is invisible, then to all intents and purposes the boundary is not there. It will be there, it will be felt, only if the crossing is marked. And essentially, the crossing of a boundary by a path can only be marked by a gateway. That is why all forms of gateway play such an important role in the environment.

A gateway can have many forms: a literal gate, a bridge, a passage between narrowly separated buildings, an avenue of trees, a gateway through a building. All of these have the same function: they mark the point where a path crosses a boundary and help

TOWNS

maintain the boundary. All of them are "things"—not merely holes or gaps, but solid entities.

Gateways mark the point of transition.

In every case, the crucial feeling which this solid thing must create is the feeling of transition.
Therefore:

Mark every boundary in the city which has important human meaning—the boundary of a building cluster, a neighborhood, a precinct—by great gateways where the major entering paths cross the boundary.

gateway

boundary

53 MAIN GATEWAYS

Make the gateways solid elements, visible from every line of approach, enclosing the paths, punching a hole through a building, creating a bridge or a sharp change of level—but above all make them "things," in just the same way specified for MAIN ENTRANCE (110), but make them larger. Whenever possible, emphasize the feeling of transition for the person passing through the gateway, by allowing change of light, or surface, view, crossing water, a change of level—ENTRANCE TRANSITION (112). In every case, treat the main gateway as the starting point of the pedestrian circulation inside the precinct—CIRCULATION REALMS (98). . . .

Figure 4.1 Overview of the *Main Gateways* pattern (Alexander et al. 1977, 276–279).

The concept of patterns was first established in *The Timeless Way of Building* (1979), and put in practice in *A Pattern Language* (1977) by the architect Christopher Alexander and his team. Their work, in the fields of architecture and urbanism, proposed a set of patterns with the purpose to empower people with the tools to design their own tows and houses. Since the publishing of this seminal work, other authors sought to expand the concept to other domains.

A Pattern Language contains 253 patterns ordered in terms or scale: from the largest ones, such as towns, until the patterns focussing on the details of construction. The sequence presented was chosen based on the connections between them, but since patterns work as a network of relations, that sequence does not represent a strict order between patterns. They work as a hierarchical system where each pattern helps to complete the ones above and is itself completed from the ones below, in short, "no pattern is an isolated entity" (Alexander et al. 1977, XIII). Patterns are also clustered in three major groups: Towns, Building, and Construction, with these being further divided in smaller groups. These groups do not have an explicit title, but rather a small description of the purpose its patterns, for example:

> *fix the position of individual buildings on the site, within the complex, one by one, according to the nature of the site, the trees, the sun: this is one of the most important moments in the language;*[20] (Alexander et al. XXVI)

Alexander's patterns consist of: a name; a ranking; a picture; the context; a short problem statement; the body of the problem; the solution; a diagram illustrating the solution; and finally, connections to other patterns. The first pattern does not have a context and the last one

20 Description for the group of patterns 104 to 109: *Site Repair*, *South Facing Outdoors*, *Positive Outdoor Space*, *Wings of Light*, *Connected Buildings*, *Long Thin House*.

does not have related patterns, therefore, reinforcing the idea that patterns in the language are interconnected in a continuous thread. The sections are not explicitly declared, that is, there are no headings delimiting each one, but the limits of a section can be inferred from the strict format used in the layout and typographic style:

- all patterns have the same number of components, and their components are always displayed in the same order;
- an ellipsis establishes the beginning of the context, another the end of the pattern — they are used to reinforce the relations between patterns, i.e., signalizing that there are patterns before and after the current one;
- the problem and solution paragraphs are formatted in bold and spatially separated from the rest of the text by a blank line;
- the solution is always introduced with "Therefore:";
- diagrams share a common graphic language;
- a series of three diamonds is used to mark the beginning of the problem, and to mark the end of the solution;
- patterns' names are always set in SMALL CAPS throughout the book, so they can be easily recognized.

Patterns are identified by a number and a name. A name is an essential component of a pattern, and expresses, in one or a few words, the problem it solves. Next to the pattern name there are a series of asterisks classifying the confidence of the authors in the pattern. A pattern is marked with two asterisks when is thought to be a 'true invariant', i.e. when the authors considered that it was not possible to properly resolve the problem stated without the use of this pattern; one asterisk when it is believed that an 'invariant' has been identified, but that it is still possible to improve the pattern; and no asterisk when the authors are certain of having not succeeded in defining a 'true invariant', i.e., when the solution presented is only one for the possibilities of solving the stated problem.

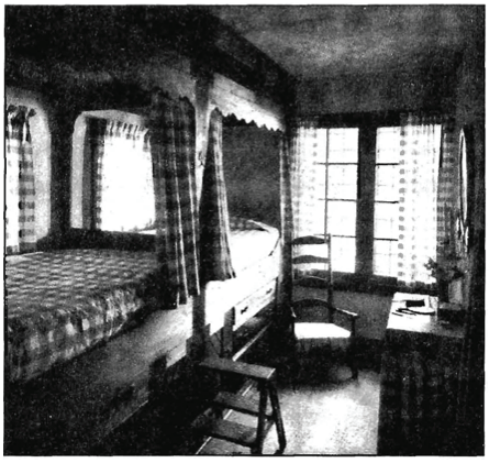


Figure 4.2 Illustration for the *Bed Cluster* pattern (Alexander et al. 1977, 676).

After the pattern's name there is an archetypal picture illustrating the application of the pattern: a photography of how the pattern's solution may work (Figure 4.2). This is followed by an introductory paragraph with the context of the pattern within the language, usually explaining how the pattern is related to larger ones. Marking the beginning of the problem there are three diamonds, followed by a paragraph, formatted in bold, stating the essence of the pattern, and summarizing the problem the pattern addresses. It follows the body of the problem, the longest section of the pattern and the place where the empirical background of the pattern is described. Here is where the forces that drive the pattern are discussed. Formatted similarly to the problem statement, the solution is stated in the form of an instruction, and presents, in a general and abstract way, what one needs to do to resolve the conflicting forces.

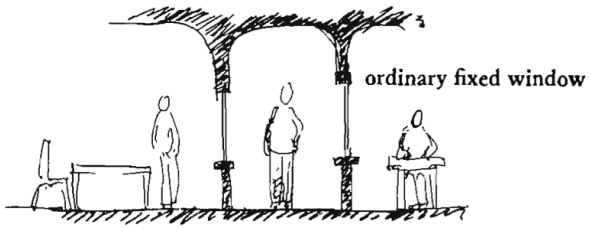


Figure 4.3 Diagram for the *Interior Windows* pattern (Alexander et al. 1977, 897).

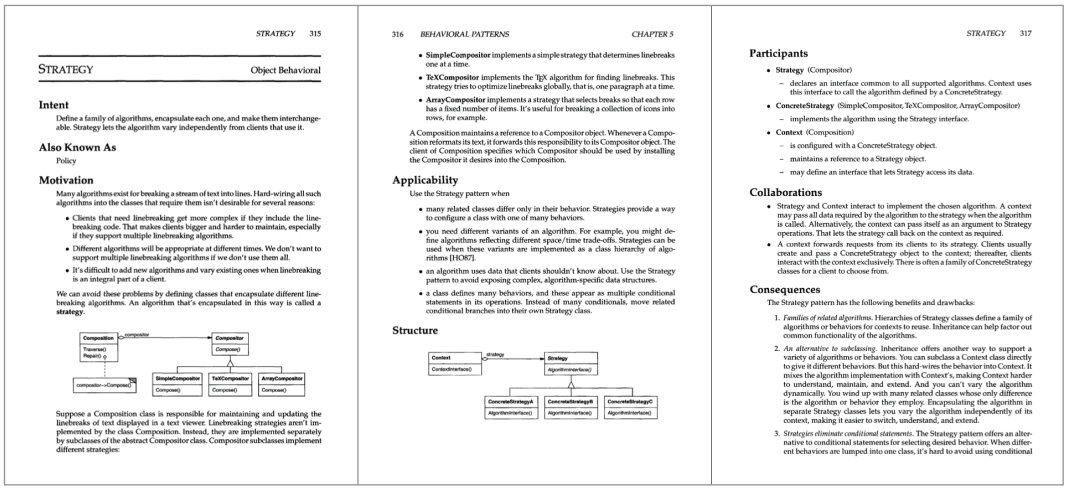
To help the reader visually understand the solution, each pattern is complemented with a sketch: a diagram (Figure 4.3) that synthesizes the solution and the spatial relations of its components. It follows the last row of three diamonds, and finally, a paragraph that connects the pattern with other patterns in the language, i.e., a list of which other patterns can be use to complete this one.

4.1.2 *Design Patterns: Elements of Reusable Object-Oriented Software*

Design Patterns: Elements of Reusable Object-Oriented Software (Gamma et al. 1995), by the authors that are also informally known as the *Gang of Four*, was the first effective attempt to expand the concept of design patterns to a domain besides architecture and urbanism. This work in the field of software engineering describes twenty-three patterns for the design of software applying the principles of object-oriented programing.

Patterns are organized in a more complex structure than that which is used in other libraries. They are simultaneously classified by two criteria: a *purpose* that reflects what the pattern does, and that can either be *Creational*, *Structural* or *Behavioral*; and a *scope* that specifies whether the pattern applies to classes or objects. That is, each pattern has a purpose and a scope, e.g., the pattern *Composite* is *Structural* and an *Object*. Authors also identified other

types of relations that a pattern can establish, such as: patterns that are used together; patterns that are alternatives; or patterns with different intents but with similar results. Although they gave an example for each case, they did not extensively developed these relations.



STRATEGY 315

STRATEGY Object Behavioral

Intent

Define a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently from clients that use it.

Also Known As

Policy

Motivation

Many algorithms exist for breaking a stream of text into lines. Hard-wiring all such algorithms into the classes that require them isn't desirable for several reasons:

- Clients that need linebreaking get more complex if they include the linebreaking code. That makes clients bigger and harder to maintain, especially if they support multiple linebreaking algorithms.
- Different algorithms will be appropriate at different times. We don't want to support multiple linebreaking algorithms if we don't use them all.
- It's difficult to add new algorithms and vary existing ones when linebreaking is an integral part of a client.

We can avoid these problems by defining classes that encapsulate different linebreaking algorithms. An algorithm that's encapsulated in this way is called a **strategy**.

Suppose a Composition class is responsible for maintaining and updating the linebreaks of text displayed in a text viewer. Linebreaking strategies aren't implemented by the class Composition. Instead, they are implemented separately by subclasses of the abstract Compositor class. Compositor subclasses implement different strategies:

316 BEHAVIORAL PATTERNS CHAPTER 5

- **SimpleCompositor** implements a simple strategy that determines linebreaks one at a time.
- **TeXCompositor** implements the TEX algorithm for finding linebreaks. This strategy tries to optimize linebreaks globally, that is, one paragraph at a time.
- **ArrayCompositor** implements a strategy that selects breaks so that each row has a fixed number of items. It's useful for breaking a collection of icons into rows, for example.

A Composition maintains a reference to a Compositor object. Whenever a Composition reformats its text, it forwards this responsibility to its Compositor object. The client of Compositions specifies which Compositor should be used by installing the Compositor it desires into the Composition.

Applicability

Use the Strategy pattern when

- many related classes differ only in their behavior. Strategies provide a way to configure a class with one of many behaviors.
- you need different variants of an algorithm. For example, you might define algorithms reflecting different space/time trade-offs. Strategies can be used when these variants are implemented as a class hierarchy of algorithms [HO87].
- an algorithm uses data that clients shouldn't know about. Use the Strategy pattern to avoid exposing complex, algorithm-specific data structures.
- a class defines many behaviors, and these appear as multiple conditional statements in its operations. Instead of many conditionals, move related conditional branches into their own Strategy class.

Structure

STRATEGY 317

Participants

- **Strategy** (Compositor)
  - declares an interface common to all supported algorithms. Context uses this interface to call the algorithm defined by a ConcreteStrategy.
- **ConcreteStrategy** (SimpleCompositor, TeXCompositor, ArrayCompositor)
  - implements the algorithm using the Strategy interface.
- **Context** (Composition)
  - is configured with a ConcreteStrategy object.
  - maintains a reference to a Strategy object.
  - may define an interface that lets Strategy access its data.

Collaborations

- Strategy and Context interact to implement the chosen algorithm. A context may pass all data required by the algorithm to the strategy when the algorithm is called. Alternatively, the context can pass itself as an argument to Strategy operations. That lets the strategy call back on the context as required.
- A context forwards requests from its clients to its strategy. Clients usually create and pass a ConcreteStrategy object to the context; thereafter, clients interact with the context exclusively. There is often a family of ConcreteStrategy classes for a client to choose from.

Consequences

The Strategy pattern has the following benefits and drawbacks:

1. *Families of related algorithms.* Hierarchies of Strategy classes define a family of algorithms or behaviors for contexts to reuse. Inheritance can help factor out common functionality of the algorithms.
2. *An alternative to subclassing.* Inheritance offers another way to support a variety of algorithms or behaviors. You can subclass a Context class directly to give it different behaviors. But this hard-wires the behavior into Context. It mixes the algorithm implementation with Context's, making Context harder to understand, maintain, and extend. And you can't vary the algorithm dynamically. You wind up with many related classes whose only difference is the algorithm or behavior they employ. Encapsulating the algorithm in separate Strategy classes lets you vary the algorithm independently of its context, making it easier to switch, understand, and extend.
3. *Strategies eliminate conditional statements.* The Strategy pattern offers an alternative to conditional statements for selecting desired behavior. When different behaviors are lumped into one class, it's hard to avoid using conditional

Figure 4.4 Overview of the *Strategy* pattern (Gamma et al. 1995, 315–317).

They also proposed an organization of the patterns based on references found in a *Related Patterns* section. The value of this data is that it makes it possible to generate a graphic representation of the relations of the patterns within the language (Figure 4.5).

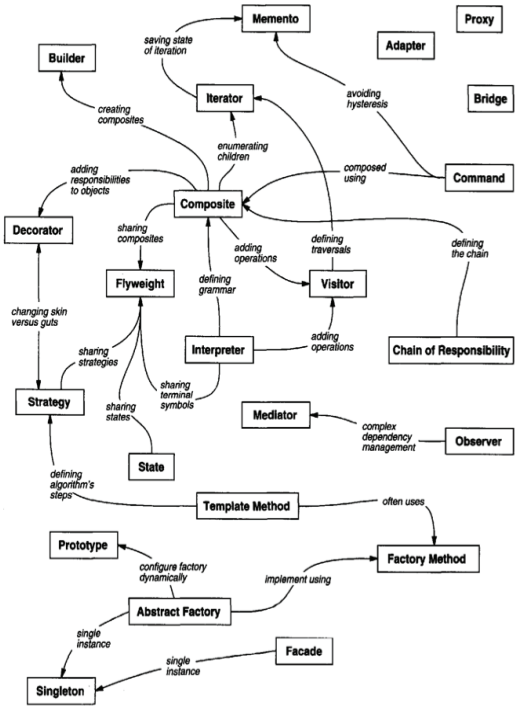


Figure 4.5 Design pattern relationships (Gamma et al. 1995, 12).

Patterns follow a consistent format to make them easier to read, use and compare. They are formatted in the following sections: a name and classification; an *Intent*; *Also Know As*; *Motivation*; *Applicability*; *Structure*; *Participants*; *Collaborations*; *Consequences*; *Implementation*; *Sample Code*; *Known Uses*; and *Related Patterns*.

The *Intent* is a short statement that explains the purpose of the pattern. The *Motivation* describes a concrete scenario that illustrates the design problem that the pattern addresses. The *Applicability* is composed as a list of situations in which one should use it, describing examples of poor design that the pattern can resolve, and how to recognize these situations. The *Structure* provides a graphical representation based on the object-modeling technique. *Participants* identifies the classes and objects that intervene in the pattern and succinctly describes their responsibilities; and *Collaborations* explains how the *Participants* work. *Consequences* enumerates the benefits and liabilities resulting from its implementation. And *Implementation* lists considerations to account for when implementing the pattern, namely, hints, techniques or language-specific issues. Patterns are illustrated by *Sample Codes* of possible implementations in C++ or Smalltalk, with explanations of those pseudo-code fragments. Then the authors offer *Known Uses* for the pattern: at least two examples from different domains, and finally, the section *Related Patterns* describes other patterns that should be used with this one, or patterns with a similar function.

4.1.3 *Common Ground: A Pattern Language for Human-Computer Interface Design*

Step-by-Step Instructions

Examples:

- Wizards
- Installation instructions for all kinds of appliances, software, etc.
- Recipes
- Repair manuals
- Getting cash from an ATM

Context: A user needs to perform a complex task, with limited time, knowledge, attention, or space. Alternatively, the nature of the task is step-by-step, and it's meaningless to show all the action possibilities at once.

Problem: How can the artifact unfold the possible actions to the user in a way that does not overwhelm or confuse them, but instead guides them to a successful task completion?

Forces:

- The user doesn't always want, or need, to understand all the details of what they are doing.
- A user presented with a bunch of possible actions, in no prescribed order, may not have any practical way to figure out what to do first, second, etc.
- A user who is afraid of doing something wrong may prefer that the actions they have to perform be explicitly spelled out for them.
- The whole task can't be performed entirely automatically, because it requires choices or information from the user.

Solution: **Walk the user through the task one step at a time, giving very clear instructions at each step.** Use visual similarities in all the steps, e.g. typography and layout, to maintain a rhythm throughout the task; make each step a focal point, both visually and in the user's "attention space." If information is needed from the user, ask for it in simple terms and with brevity; by keeping it short, you can better maintain the user's sense of flow through the whole step-by-step process.

The task may branch like a flow chart, depending upon what information the user gives it, but the user doesn't necessarily need to know about all the available paths through the task. If it's not likely to confuse the user, show the steps as a Map of Navigable Spaces. If possible, allow the user to back out of the steps (Go Back One Step, Go Back to a Safe Place). If the sequence of steps as seen by the user is too long -- more than ten steps, for example -- try to break it up into manageable sub-sequences, so it doesn't get too tedious for the user. Make sure the sub-sequences relate to each other in a meaningful way, however, or the user may see it as gratuitous or annoying.

Sometimes users may want to know more about what they're doing -- Optional Detail On Demand gives you a way to present that extra information. Also, if a user has gone through a lot of steps, they have trouble remembering what they've done and why. At least provide a Progress Indicator if the number of steps grows beyond seven or eight, which is the average limit of short-term memory. If a lot of user interaction is necessary, such as for branching decisions, consider providing an Interaction History

Resulting Context: Narrative is a good choice for presenting the task steps themselves; the use of natural language to describe what needs to be done is intuitively obvious, and puts a user at ease. Go Back One Step and Go Back to a Safe Place, along with a corresponding Forward control, can be used to move through an interactive task.

To get information from the user, you can use a Form or its simpler component patterns, especially Choice from a Small Set and Forgiving Text Entry. Using Good Defaults with them allows the user to move smoothly past the points where extra data entry is unnecessary, again preserving the sense of flow. Finally, a small set of Convenient Environment Actions should give the user ways to cancel or suspend the task without having to back out of it one step at a time.

Notes: Be aware that this pattern may irritate experienced users. If a user knows exactly what they need to do, and want to do it quickly, constraint to this step-by-step presentation can feel like a straitjacket! Also, if the task to be accomplished isn't inherently linear -- i.e. you don't really have to do one step first, another step second, etc. -- you might provide an alternative "random access" presentation of the possible actions, such as a Stack of Working Surfaces.

Figure 4.6 Overview of the *Step-by-Step Instructions* pattern (Tidwell 1999).

Common Ground: A Pattern Language for Human-Computer Interface Design (Tidwell 1999) was the first major effort to develop a pattern library dedicated to interaction design. This work was initially presented at the *PLoP'98* — conference on pattern language of programs — and it is currently available on the web.[21] The author presents sixty patterns for the design of interfaces, essentially computer applications and websites; of which six are unwritten i.e., only the name is expressed.

Patterns are arranged by groups, in which the group title is posed as a question, e.g., "What specific actions should the user take?" and are disclosed in a top-down manner: from the structural elements, until the details. The first two groups of patterns represent what the author calls "primary patterns": patterns that form the 'backbone' of the language, and define and constrain the shape of an interface. The first group is composed by the patterns that form the basic shape of the content; the second group contains patterns that form the basic shape of the actions taken by the user with the interface.

In addition to the main pattern language, the author proposes several sublanguages based on the primary patterns defined earlier. These sublanguages do not form in any way a strict organization of the language, but are rather suggestions of patterns that might work well together as a complement to the primary patterns.

In short, a pattern is composed in the following sections: a name; *Examples*; the *Context*; the *Problem*; *Forces*; the *Solution*; the *Resulting Context*. Contrary to other libraries, the examples provided are only textual, and are composed as a list of cases where the pattern is used; examples can be generic (*Spreadsheets*), or about a concrete application that uses the pattern (*Mac bubble-help*). Some offer bad examples, sometimes showing inappropriate uses of a pattern, others times showing situations in which the patterns should be used but was not. These bad examples are equivalent to anti-patterns, though there are not described as such.

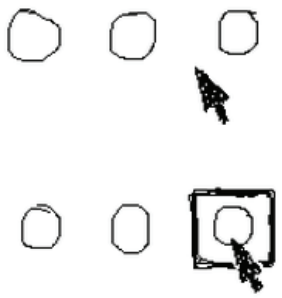


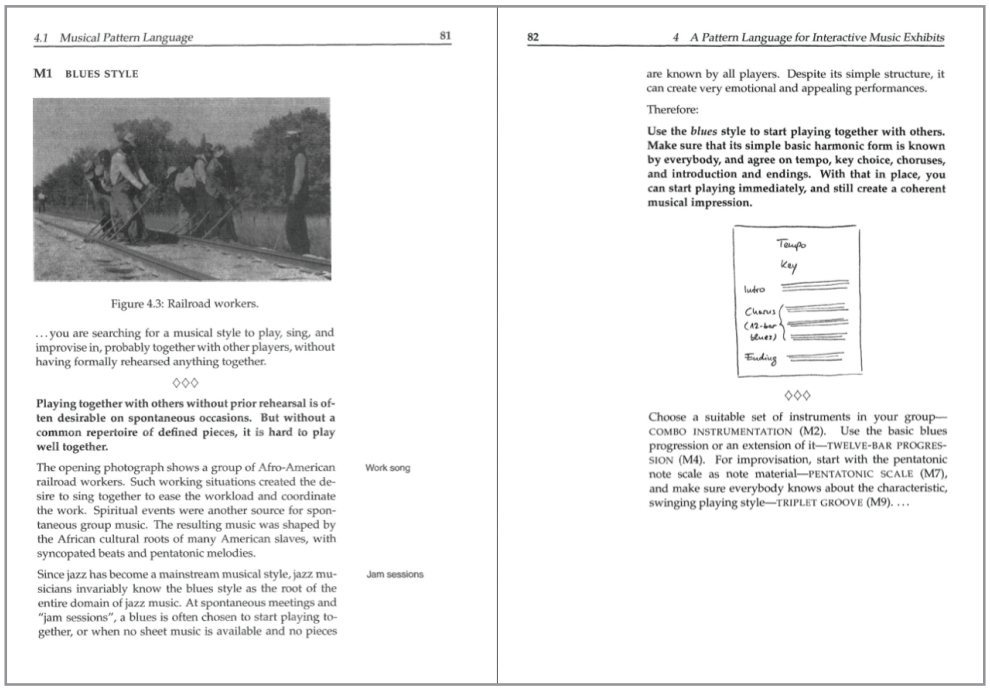
Figure 4.7 Illustration for the *Pointer Shows Affordance* pattern (Tidwell 1999).

The *Context* is composed in a paragraph stating the situations in which the pattern should be used, usually referencing patterns for which this one is a complement. The *Problem* is stated

21 www.mit.edu/~jtidwell/interaction_patterns.html

as a question, e.g., "How should the information be organized?". *Forces* are composed as a list, expressing the aspects that shape the pattern. The *Solution* is composed by: a prescriptive sentence, formatted in bold, expressing how to implement the pattern, e.g., "Show the data in a tree-like structure."; and a paragraph with a longer explanation. A few patterns are complemented by a drawing (*Pointer Shows Affordance*, Figure 4.7) or a screenshot (*Progress Indicator*). The *Resulting Context*, in contrast with the *Context*, expresses the new context that appeared with the use of the pattern, proposing new ones that might complement it.

4.1.4 *A Pattern Approach to Interaction Design*



4.1 Musical Pattern Language 81

M1 BLUES STYLE

Figure 4.3: Railroad workers.

. . . you are searching for a musical style to play, sing, and improvise in, probably together with other players, without having formally rehearsed anything together.

◊◊◊

Playing together with others without prior rehearsal is often desirable on spontaneous occasions. But without a common repertoire of defined pieces, it is hard to play well together.

Work song

The opening photograph shows a group of Afro-American railroad workers. Such working situations created the desire to sing together to ease the workload and coordinate the work. Spiritual events were another source for spontaneous group music. The resulting music was shaped by the African cultural roots of many American slaves, with syncopated beats and pentatonic melodies.

Jam sessions

Since jazz has become a mainstream musical style, jazz musicians invariably know the blues style as the root of the entire domain of jazz music. At spontaneous meetings and "jam sessions", a blues is often chosen to start playing together, or when no sheet music is available and no pieces

82 4 A Pattern Language for Interactive Music Exhibits

are known by all players. Despite its simple structure, it can create very emotional and appealing performances.

Therefore:

Use the *blues* style to start playing together with others. Make sure that its simple basic harmonic form is known by everybody, and agree on tempo, key choice, choruses, and introduction and endings. With that in place, you can start playing immediately, and still create a coherent musical impression.

◊◊◊

Choose a suitable set of instruments in your group—COMBO INSTRUMENTATION (M2). Use the basic blues progression or an extension of it—TWELVE-BAR PROGRESSION (M4). For improvisation, start with the pentatonic note scale as note material—PENTATONIC SCALE (M7), and make sure everybody knows about the characteristic, swinging playing style—TRIPLET GROOVE (M9). . . .

Figure 4.8 Overview of the *Blues Style* pattern (Borchers 2001, 81–82).

The patterns presented in *A Pattern Approach to Interaction Design* (Borchers 2001) are the result of the experience gained by the author in the course of four projects: the *WorldBeat*, an interactive exhibit; the *Interactive Fugue* exhibit; the *Personal Orchestra*, a virtual orchestra; and the *Virtual Vienna*, a virtual city tour of Vienna.

The outcome of those four projects were three pattern languages, each one for a different domain: *Musical Pattern Language*; *HCI Pattern Language*; and *Software Pattern Language*. The first is a pattern language for blues music; the second is for interactive systems that are used in public places, namely, interactive exhibits at museums and exhibit centers such as kiosks; the last one is a language for interactive music software. Each language is summarized in a graph — similar to that used by the *Gang of Four* (Figure 4.5) — with all the relations established between patterns, that provides a quick overview for the language (Figure 4.9).

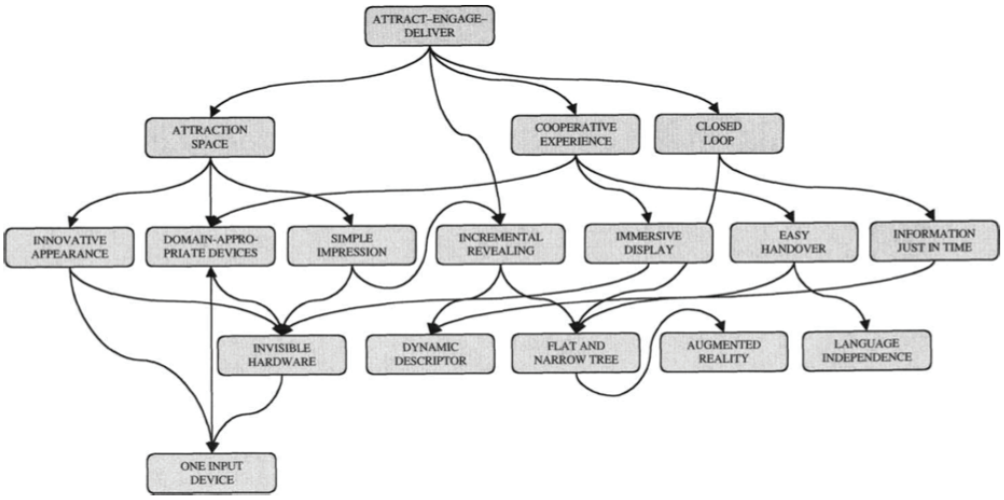


Figure 4.9 Design patterns relationships for the *HCI Pattern Language* (Borchers 2001, 104).

Borchers uses a structure very similar to the one used in Alexander's work, with the same organization and layout, formatted in the following sections: a name; a ranking; an illustration — depending on the context, it may be a photograph of the system being used (Figure 4.10), or a screenshot; the problem; forces; additional examples of the real-world application of the pattern; the solution; the diagram; and related patterns.

Figure 4.10 Illustration for the *Easy Handover* pattern (Borchers 2001, 117)

Patterns are identified by a name and an alphanumeric code: a letter representing the corresponding language (e.g., a "S" represents the *Software Pattern Language*); and a number identifies the pattern in the language. The name and the corresponding code are always used when a pattern is referenced in another section of the book.

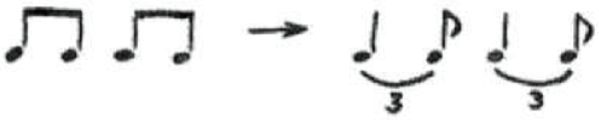


Figure 4.11 Diagram for the *Triplet Groove* pattern (Borchers 2001, 98).

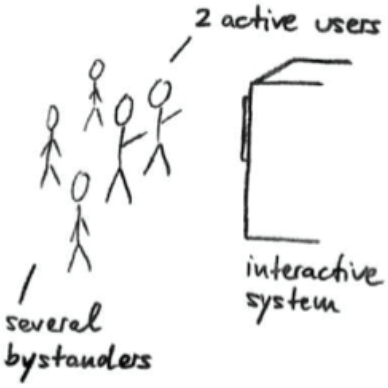


Figure 4.12 Diagram for the *Cooperative Experience* pattern (Borchers 2001, 113).

Diagrams are drawn in different graphical styles, depending the domain of the pattern. For example, music related patterns use musical score notations (Figure 4.11); HCI patterns have a drawing of the installation schematics (Figure 4.12); and software patterns use organizational charts.

4.1.5 *The Design of Sites: Patterns for Creating Winning Websites*

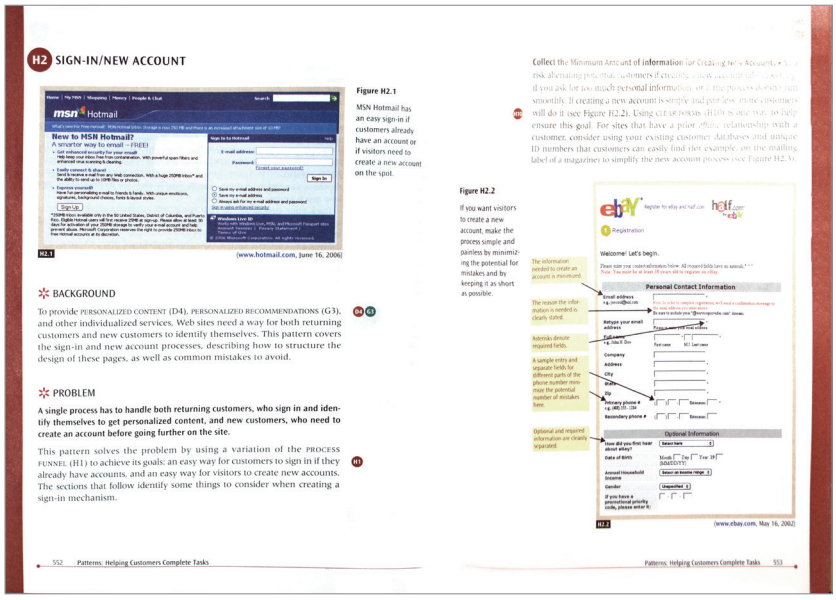


Figure 4.13 Overview of the *Sign-in/New Acount* pattern (van Duyne, Landay and Hong 2006).

The Design of Sites (van Duyne, Landay and Hong 2006) is the second edition of a book first published in 2003 that contains patterns for the design of websites. In addition to the book there is a supporting website,[22] which functions primarily as a promotional website, where we can find simplified versions of the patterns in the book.

They are organized in thirteen groups of thematically related patterns (e.g., *Basic E-Commerce*) which are identified by a name, a letter, and a color. The disposal of these groups is not arbitrary, it starts with the more broad ones until those that deal with details (e.g., *Speeding Up Your Site*), or as the authors put it, "the earlier the pattern group appears on this scheme, the earlier it should be used in the design process" (31).

Patterns are formally organized in six parts: name, background, problem, forces, solution, and other patterns to consider; these sections are explicitly titled, except for the forces.

The name of each pattern was chosen so that it consists of a phrase that can be used within a sentence, e.g., "You should add *Category Pages* to this site". As in *A Pattern Language*, patterns' names used throughout the book are formatted in small caps to be quickly identify. Besides the name, each pattern is also identified by a unique code, consisting of a letter and a number inside a color-coded circle, in which, the color and the letter identify the group of the pattern; and the number, the pattern within the group. When a pattern is referenced in the text, this code shows up on the text margins, and the textual part of this code shows in a parenthesis next to the name.

Following the name there is an image illustrating the pattern (with a corresponding label) describing the image, and the date in which this screenshot was taken. Next, there is a background providing the pattern context: other patterns that originate this one, and the scope of the pattern. It is followed by the problem (formatted in bold), which consists of a succinct description of the specific problem that this pattern addresses.

The longest part of the pattern is the one dedicated to its forces, and depending on the pattern, it may be divided in multiple sections, addressing the different questions that one must answer to solve a design problem. This text is formed primarily by text in prose, but can have additional images or graphics.

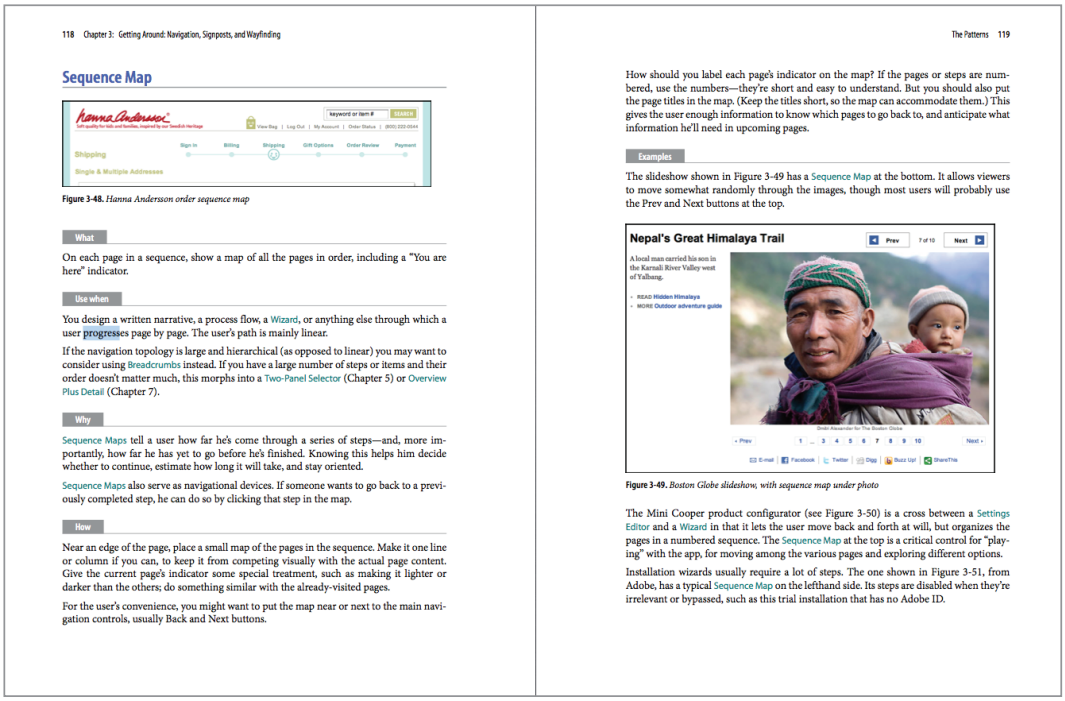
Next comes the solution — formally similar to the problem, — formatted in bold and consisting of a succinct statement of how to solve the problem; accompanied by a sketch of the pattern. Finally, there are recommendations of additional patterns that may help to complete the current one.

Although this book was published in 2006, the last group of patterns offers a small sample of patterns for the design of sites in mobile devices: *Mobile Screen Sizing*, *Mobile Input Controls*, and *Location-based Services*. While these patterns do not necessarily represent the character-

22 www.designofsites.com

istics of the current mobile websites, and are optimized for what is now more commonly described as feature phones, it is still interesting to take notice of them.

4.1.6 *Designing Interfaces*



118 Chapter 3: Getting Around: Navigation, Signposts, and Wayfinding

Sequence Map

Figure 3-48. *Hanna Andersson order sequence map*

What

On each page in a sequence, show a map of all the pages in order, including a "You are here" indicator.

Use when

You design a written narrative, a process flow, a Wizard, or anything else through which a user progresses page by page. The user's path is mainly linear.

If the navigation topology is large and hierarchical (as opposed to linear) you may want to consider using Breadcrumbs instead. If you have a large number of steps or items and their order doesn't matter much, this morphs into a Two-Panel Selector (Chapter 5) or Overview Plus Detail (Chapter 7).

Why

Sequence Maps tell a user how far he's come through a series of steps—and, more importantly, how far he has yet to go before he's finished. Knowing this helps him decide whether to continue, estimate how long it will take, and stay oriented.

Sequence Maps also serve as navigational devices. If someone wants to go back to a previously completed step, he can do so by clicking that step in the map.

How

Near an edge of the page, place a small map of the pages in the sequence. Make it one line or column if you can, to keep it from competing visually with the actual page content. Give the current page's indicator some special treatment, such as making it lighter or darker than the others; do something similar with the already-visited pages.

For the user's convenience, you might want to put the map near or next to the main navigation controls, usually Back and Next buttons.

The Patterns 119

How should you label each page's indicator on the map? If the pages or steps are numbered, use the numbers—they're short and easy to understand. But you should also put the page titles in the map. (Keep the titles short, so the map can accommodate them.) This gives the user enough information to know which pages to go back to, and anticipate what information he'll need in upcoming pages.

Examples

The slideshow shown in Figure 3-49 has a Sequence Map at the bottom. It allows viewers to move somewhat randomly through the images, though most users will probably use the Prev and Next buttons at the top.

Nepal's Great Himalaya Trail

Figure 3-49. *Boston Globe slideshow, with sequence map under photo*

The Mini Cooper product configurator (see Figure 3-50) is a cross between a Settings Editor and a Wizard in that it lets the user move back and forth at will, but organizes the pages in a numbered sequence. The Sequence Map at the top is a critical control for "playing" with the app, for moving among the various pages and exploring different options.

Installation wizards usually require a lot of steps. The one shown in Figure 3-51, from Adobe, has a typical Sequence Map on the lefthand side. Its steps are disabled when they're irrelevant or bypassed, such as this trial installation that has no Adobe ID.

Figure 4.14 Overview of the *Sequence Map* pattern (Tidwell 2011, 118–119).

Designing Interfaces (Tidwell 2005) is the successor of Tidwell's previous work, in which it was proposed a first set of pattern for the design of interactive systems. This work offers 125 patterns for the design of interfaces, in particular, applications and websites. In the second edition of the book (Tidwell 2011) besides updated patterns and examples, the most relevant addition is a chapter dedicated to mobile interfaces.

Patterns are organized in eleven chapters, loosely ordered in terms of scale, implying some sort of top-down hierarchy, and corresponding to the order that they usually appear in the design process. It starts with the users' behavior, and then follows the organization of the content until the details.

All patterns follow a similar structure except the ones in first chapter, because these patterns are related to the users' behavior with interactive systems rather than graphic components. Thus, these patterns only have a name; a subtitle, which is a sentence posed as a hypothetical opinion of a user about the pattern, for example, the *Keyboard Only* has as subtitle "Please don't make me use the keyboard."; and the reasoning of the pattern.

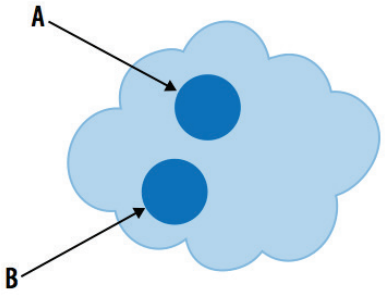


Figure 4.15 Illustration for the *Clear Entry Points* pattern (Tidwell 2011, 87).

The consequent patterns are composed by a name that is styled throughout the book in blue and capitalized. This is followed by an illustration: usually a screenshot from a website or application; when the pattern does not have a clear shape, a schematic representation is used instead, e.g., *Clear Entry Points* (Figure 4.15). The explanation for the pattern is distributed through four sections: *What*, *Use When*, *Why*, *How*. It is also provided a section with additional *Examples* or implementation details, and another section with links to equivalent patterns *In other libraries*.

4.1.7 *Yahoo! Design Pattern Library*

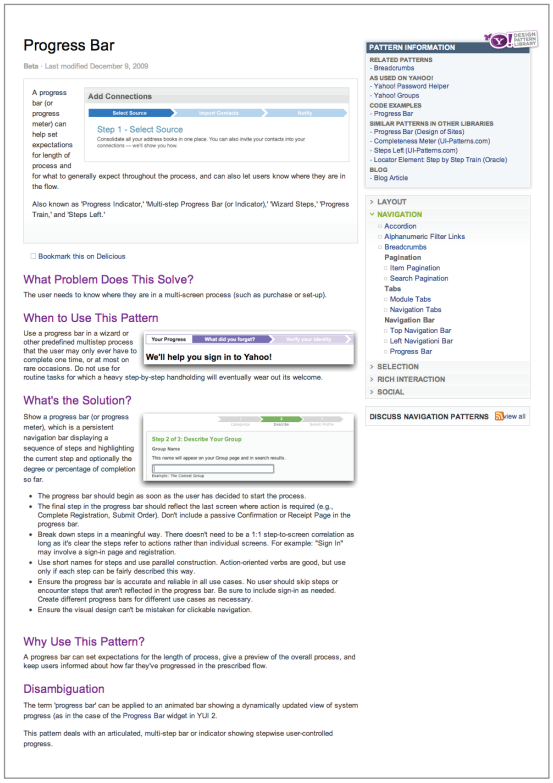


Figure 4.16 Overview of the *Progress Bar* pattern (Yahoo! 2006).

Developed by *Yahoo!* and published in 2006,[23] this library currently contains 59 patterns that are used on Yahoo!'s own network of sites.

Patterns are organized in categories, with five groups in the top level: *Layout*, *Navigation*, *Selection*, *Rich Interaction*, and *Social*. As other libraries, this one also tries to organize and sort the main categories in terms of scale; sub-levels are organized by the pattern function, e.g., all types of transition are in the same section. One difference between this and other libraries analyzed is that, here, patterns can appear at any level, except at the first one, i.e., we can find a pattern next to a section name, being both at the same level. This is also the library with most hierarchical levels: a total of four.

Each pattern starts with a title, ranking, last modification and a major example (always from the *Yahoo!* network), followed by a small description. The main content is divided in multiple sections, addressing four central questions: *What Problem Does This Solve?*, *When to Use This Pattern?*, *What's the Solution?*, *Why Use This Pattern?* Besides these main sections, other optional sections may be developed, such as: *Accessibility* or *Special Cases*. Almost all the textual content of the patterns is composed as lists, with small paragraphs, and occasionally with images providing additional details about its implementation; however, these images are not directly linked to the corresponding website, instead, links are aggregated on a list in the sidebar, which makes it more difficult to find the correct example. Finally, additional examples of the pattern in other websites may also be offered.

Beyond the pattern main components, associated content may be provided in a sidebar: *related patterns* to accomplish the same objective; *code examples* or instructions of how to technically implement each pattern; where can we find the pattern on the *Yahoo!* network; where can we find it on external websites; wireframe stencils. The library is also open to users' feedback and discussion through a forum, but participation can be considered poor.[24]

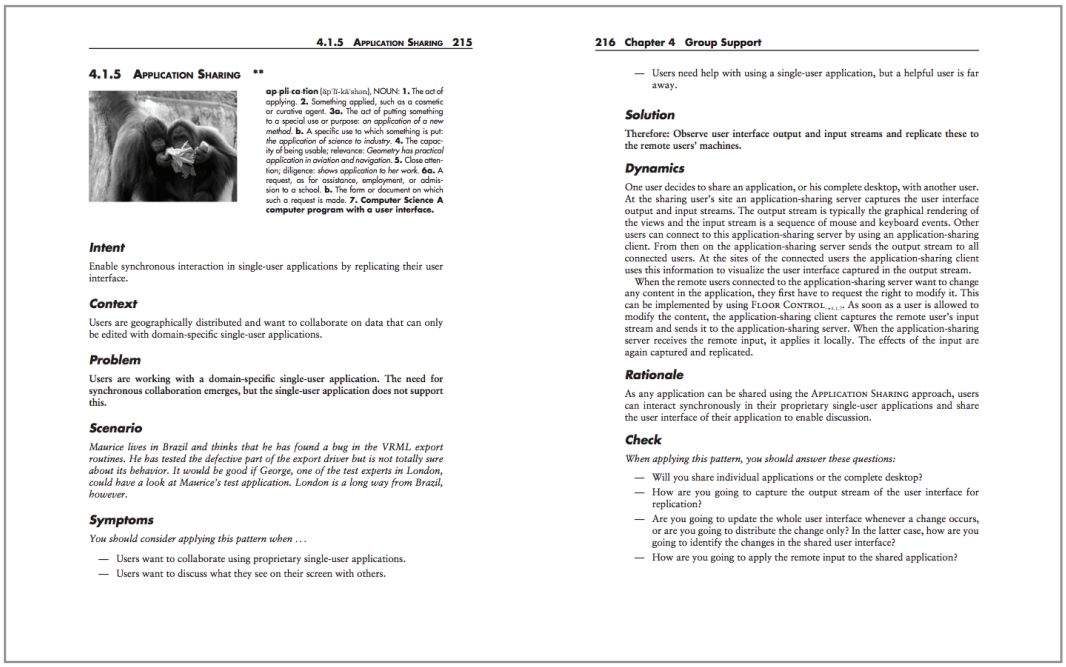
One interesting aspect is how ratings are presented. They are not expressed in a numerical scale, but rather textually: *Beta*; *Working Solution*; *Best Practice*, but we can still translate those expressions to a three-level scale, as in Alexander's patterns. Because of this less abstract approach, it is easier to decode the meaning of the rating; however, because of the expressions chosen, it is also easier to misunderstand its purpose, for example, we could understand *Best Practice* as a pattern we must always use, though, that is not what was intended.

---

23 As stated by Erin Malone (Crumlish 2009), *Yahoo!* started the development of this pattern library in 2004, only later, in 2006 it was decided to publish a subset of the library to the public.

24 Inside the *Yahoo! Developer Network* there is a forum dedicated to design patterns, and within, a subforum for each major section (e.g., Social). Since its opening in 2008, around twenty topics were started, many of them with not much more than two posts, most made by the same Yahoo! employee; and with some of the topics unanswered.

4.1.8 *Patterns for Computer-Mediated Interaction*



4.1.5 Application Sharing 215

4.1.5 Application Sharing **

ap pli·ca·tion [ăp'lĭ-kā'shən], NOUN: **1.** The act of applying. **2.** Something applied, such as a cosmetic or curative agent. **3a.** The act of putting something to a special use or purpose: *an application of a new method*. **b.** A specific use to which something is put: *the application of science to industry*. **4.** The capacity of being usable; relevance: *Geometry has practical application in aviation and navigation*. **5.** Close attention; diligence: *shows application to her work*. **6a.** A request, as for assistance, employment, or admission to a school. **b.** The form or document on which such a request is made. **7. Computer Science A computer program with a user interface.**

Intent

Enable synchronous interaction in single-user applications by replicating their user interface.

Context

Users are geographically distributed and want to collaborate on data that can only be edited with domain-specific single-user applications.

Problem

Users are working with a domain-specific single-user application. The need for synchronous collaboration emerges, but the single-user application does not support this.

Scenario

Maurice lives in Brazil and thinks that he has found a bug in the VRML export routines. He has tested the defective part of the export driver but is not totally sure about its behavior. It would be good if George, one of the test experts in London, could have a look at Maurice's test application. London is a long way from Brazil, however.

Symptoms

You should consider applying this pattern when . . .

— Users want to collaborate using proprietary single-user applications.
— Users want to discuss what they see on their screen with others.

216 Chapter 4 Group Support

— Users need help with using a single-user application, but a helpful user is far away.

Solution

Therefore: Observe user interface output and input streams and replicate these to the remote users' machines.

Dynamics

One user decides to share an application, or his complete desktop, with another user. At the sharing user's site an application-sharing server captures the user interface output and input streams. The output stream is typically the graphical rendering of the views and the input stream is a sequence of mouse and keyboard events. Other users can connect to this application-sharing server by using an application-sharing client. From then on the application-sharing server sends the output stream to all connected users. At the sites of the connected users the application-sharing client uses this information to visualize the user interface captured in the output stream.

When the remote users connected to the application-sharing server want to change any content in the application, they first have to request the right to modify it. This can be implemented by using Floor Control. . . . As soon as a user is allowed to modify the content, the application-sharing client captures the remote user's input stream and sends it to the application-sharing server. When the application-sharing server receives the remote input, it applies it locally. The effects of the input are again captured and replicated.

Rationale

As any application can be shared using the Application Sharing approach, users can interact synchronously in their proprietary single-user applications and share the user interface of their application to enable discussion.

Check

When applying this pattern, you should answer these questions:

— Will you share individual applications or the complete desktop?
— How are you going to capture the output stream of the user interface for replication?
— Are you going to update the whole user interface whenever a change occurs, or are you going to distribute the change only? In the latter case, how are you going to identify the changes in the shared user interface?
— How are you going to apply the remote input to the shared application?

Figure 4.17 Overview of the *Application Sharing* pattern (Schümmer and Lukosch 2007, 215–216).

Published in 2007 by Till Schümmer and Stephan Lukosch, this book discusses patterns for applications supporting computer-mediated interactions or groupware applications.

Patterns in this work are arranged in what the authors call 'layers', each one addressing a different problem. In the top layer we find "patterns for establishing a community" (Schümmer and Lukosch 2007, 43), in the second "patterns for supporting small groups in their interaction" (44) and the lowest layer "patterns for designing the infrastructure that is needed by the groupware tools" (44). Each layer is further divided in 'clusters', each one addressing a specific theme or type of need.

In short, each pattern contains: a name; a 'sensitizing' photograph; the *Intent*; the *Context*; the *Problem*; a *Scenario*; *Symptoms*; the *Solution*; the *Dynamics*; the *Rationale*; a *Check List*; *Danger Spots*; *Known Uses*; and, *Related Patterns*.

Patterns start with a name that should be memorizable so it can be used in communication within a project (30). Throughout the book, names are formatted in small caps and assigned a corresponding index number (e.g., 3.1.7) representing the chapter and section where the pattern can be found. The name is complemented with alternative names by which the pattern may be known. Next to the name is the pattern ranking. Authors opted to use the same type of ranking used in Alexander's patterns: two asterisks, one asterisk or none.

Then, follows a 'sensitizing' picture illustrating the pattern, a picture that should capture the essence of the patterns and help users remember it (30). However, oddly enough, this im-

age is not necessarily an explicit representation of the pattern, but rather a metaphor of the pattern solution. Authors give the example of the *Activity Log* pattern which is illustrated by a photograph of an elephant's head, meaning that this patterns advocates remembering activities like elephants (Figure 4.18). The picture is further complemented with a dictionary definition of some part of the pattern name; for example, in the case of the *Activity Log* there is the definition of the word log.

log (lôg), NOUN: **1a.** A usually large section of a trunk or limb of a fallen or felled tree. **b.** A long thick section of trimmed, unhewn timber. **2. Nautical a.** A device trailed from a ship to determine its speed through the water. **b. A record of a ship's speed, its progress, and any shipboard events of navigational importance. c. The book in which this record is kept.** **3.** A record of a vehicle's performance, as the flight record of an aircraft. **4.** A record, as of the performance of a machine or the progress of an undertaking: *a computer log; a trip log.*

Figure 4.18 Illustration for the *Activity Log* Pattern (Schümmer and Lukosch 2007, 371).

The main body of the pattern follows, starting in the *Intent*, which states the purpose of the pattern in a prescriptive sentence, e.g., "Show who has been active at a specific point in time." (104) Afterward, there is the pattern's *Context*, where the authors describe the situation for which the pattern was intended, followed by the description of the problem, which gives a quick overview of the problem the pattern tries to solve; authors recommend that one should start reading a pattern from this section. To help in the contextualization of the pattern, there is a section describing a possible scenario for its application: a description of a concrete example of the pattern use. In the beginning of the book the authors tell the story of Paul Smith, a software engineer who works in a collaborative game engine development community, whose story is the main setting for the scenarios described in each pattern.

Then, there is a list of *Symptoms*, presenting the situations when one should consider the use of this pattern.

This is followed by a paragraph with the solution, which succinctly provides instructions on how to implement the pattern, and it is formatted similarly to the corresponding section in *A Pattern Language* (Alexander et al. 1977): a paragraph styled in bold, and starting with "Therefore:".

Then appears the main explanation for the pattern. First, in the section *Dynamics*, it is discussed the elements that are in play when this pattern is used; then the *Rationale* with the reason of why we should use this pattern. Followed by a section named *Check*, which offers a list of questions we must answer when applying it. And then, the *Danger spots* presents new conflicts that may arise when in this pattern is used.

Examples of the implementation of the patterns are present in the section *Known Uses*. The majority of these are textual descriptions of sites, applications or other types of systems that implemented the pattern, many are complemented by screenshots.

Finally, there is the *Related Patterns* section, describing the patterns that may complement this one. Generally the connections are established to other patterns in this library, but they can also reference external libraries, for example, the pattern *Active Map* lists as a complement to it, the pattern *Annotated Scrollbar* present in *Designing Interfaces* (Tidwell 2011).

4.1.9 *Info Design Patterns*

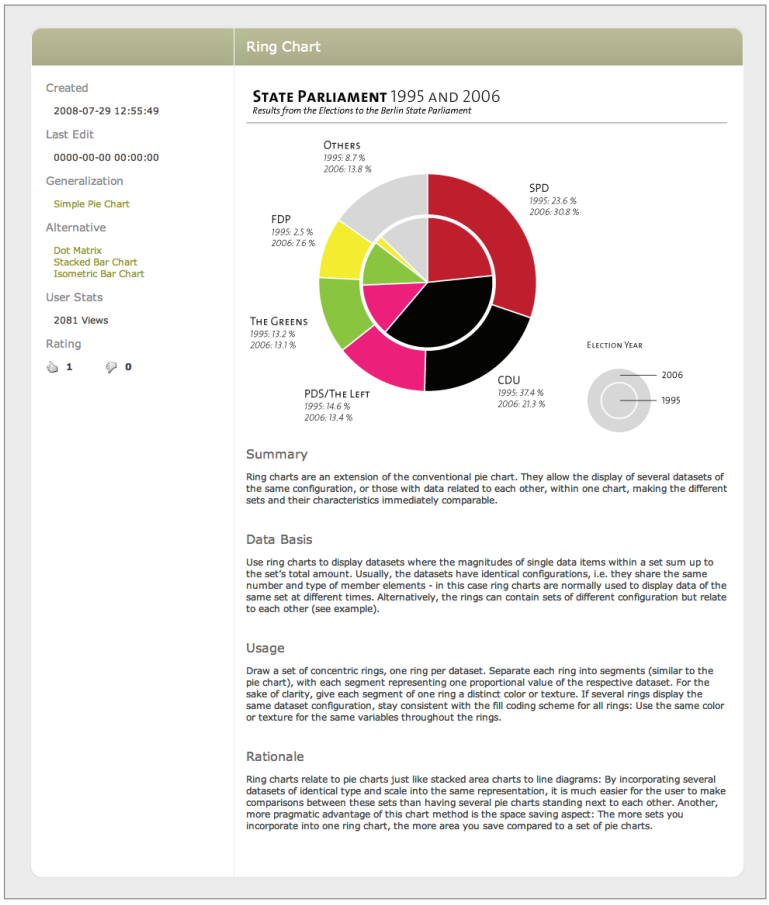


Figure 4.19 Overview of the *Ring Chart* pattern (Behrens 2008b).

The Form of Facts and Figures (2008a) is the title of the Master's thesis of Christian Behrens, a work with design patterns for information visualization. His print work is accompanied by a website — *Info Design Patterns*[25] (2008b) — only with the patterns of his thesis.

---

25 www.infodesignpatterns.com — The current version of the site is different from the one that was available at the time of writing of his document.

Patterns are organized hierarchically in three major categories (grouped by purpose), and divided further in subcategories, although in the website version the top-level is not displayed. Additionally to the taxonomy used, Behrens also proposes the use of a tag system in the online version[26] of his work, for the reason that within a hierarchic system "the rigid tree structure proves somewhat inflexible the larger the collection grows" (Behrens 2008a, 43) Therefore, each pattern has one or more labels assigned to it that attempt to summarize its characteristics, and allow users, for example, to search for a pattern that matches two conditions.

Each pattern can be identified by a code, composed by an alphanumeric ID that precedes the pattern title, in which a letter denotes the pattern major category, and the two-part number represents the subcategory and the pattern. Besides the title and identification code, each one is also associated with an iconic image visible next to its title.

Patterns were designed to be able to perform a certain task autonomously, but also to work inside a system of patterns where interconnections are established among them. Thus, the author devised a system of relations where a pattern could be connected at the same time to multiple patterns. Five types of relations, each one with an associated icon, were created:

- A pattern is called *Specialization* when it shares the same functionality of another, but has more specialized characteristics or features — analogous to the concept of inheritance in software development.
- Consequently, a pattern is a *Generalization* of another when it serves a more generic purpose. This and the former type of relation are bi-directional, i.e. when a pattern is a *Specialization*, the opposite pattern is a *Generalization*.
- A pattern is called an *Alternative* when a different pattern can be used to archive the same or similar purpose.
- Patterns can be linked by *Collaboration* when the use of a pattern depends on some functionalities of another.
- Finally, *Feature Similarity* describes when a pattern shares some functional aspects, but not enough to be qualified and recommend as an alternative.

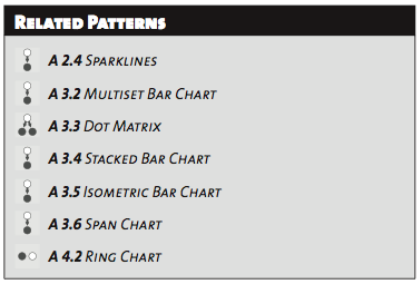


Figure 4.20 Summary of the relations that the *Simple Bar Chart* pattern can perform (Behrens 2008a, 67).

26 The tags that he mentions in his thesis are currently absent from the website, thus, it was not possible to analyze them in more detail.

In the website these related patterns are linked on the sidebar, while on the print version, this information is visible in two places: on the first spread of each pattern description, in a row of bookmarks attached to the right margin, using only its code ID; at the end of each pattern with its full title and the correspondent icon (Figure 4.20).

The pattern itself consists of four sections: the *Description*, which gives a short introduction on its purposes and outlines the problems that usually trigger its usage; the *Required Data* describing what kind of data is needed to use the pattern; the *Used* section outlines the steps we have to perform to implement it; and the *Rationale* summarizes the pattern and arguments it for its usage, i.e. why it is a good idea to use it.

Patterns are illustrated by an infographic, with the exception of a few, e.g., *Faceted Browser*. The author opted to develop a consistent graphic language common to all patterns, so users would not be distracted by structural or functional details. In addition to the infographic, each pattern is illustrated by a 'real-world' example of the application of the pattern; however, in the online version this image does not exist. Images also have a label explaining what kind of data it is representing, and from where the image was taken.

4.1.10 *Patternry*

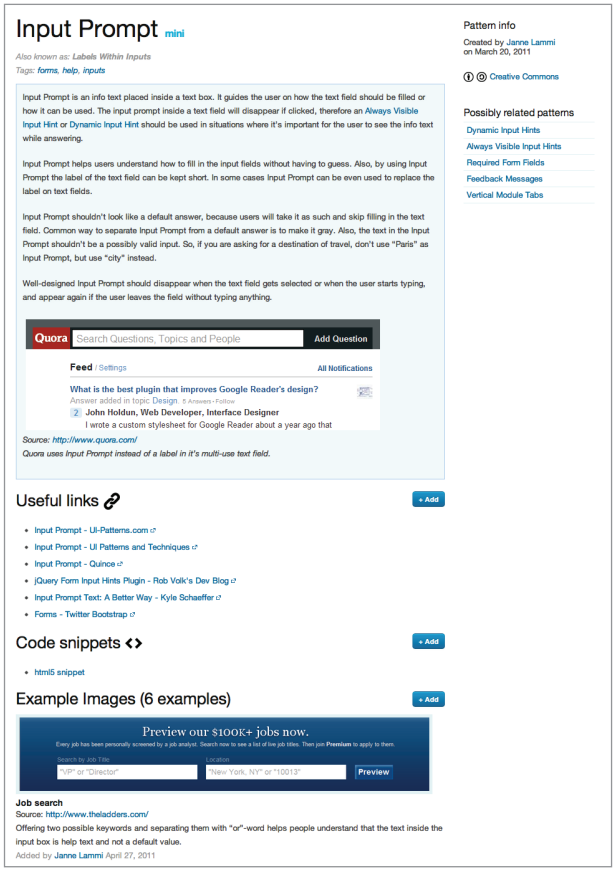


Figure 4.21 Overview of the *Vertical Module Tabs* pattern (Pattern Factory 2009).

Patternry[27] is a commercial service focused on delivering a way to document, share and collaborate on design patterns. It offers a public pattern library — *Patternry Open Library* — as a demonstration of some of its functionalities. In short, with this service users are able to create their own private pattern libraries and share them with coworkers.

The public library presents thirty-seven patterns, seven of which are marked as *mini*, as they exhibit a lower level of development. Since the main focus of this service is in allowing users to create their own private libraries, they cannot edit the main body of the public patterns, or freely add new ones; however, users are free to add new examples, code snippets, useful links, and comments.

Patterns are displayed without any hierarchy, i.e., they are all at the same level. However, some methods are implemented to help on its organization: patterns can be sorted alphabetically or by creation date, and filtered by tags.

Patterns are identified by a name, but alternative names (*Also known as*) of how they might be known are also provided. Date and author of the last edit are also displayed. A major difference on the formatting of these patterns is that the solution is presented before its explanation. A visually highlighted block shows the solution next to an illustration of the pattern, generally is used a single screenshot, but when the pattern deals with more dynamic content, a video (e.g., *Inline Edit*) or a series of screenshots (e.g., *Ratings*) may be used. Only then follows a more extensive explanation of the pattern, which addresses four main topics: what, when, how, why;[28] some patterns may have one section dedicated to accessibility, and another to sources — where we can find this pattern on other libraries, or articles that inspired it; however, in the *mini* patterns this extensive explanation is missing. Nearly all of the text is composed as lists, except for the first block. Finally, there is a section with *Possibly related patterns*, composed with patterns (programmatically chosen) that share a common tag.

4.1.11 *Designing Web Interfaces*

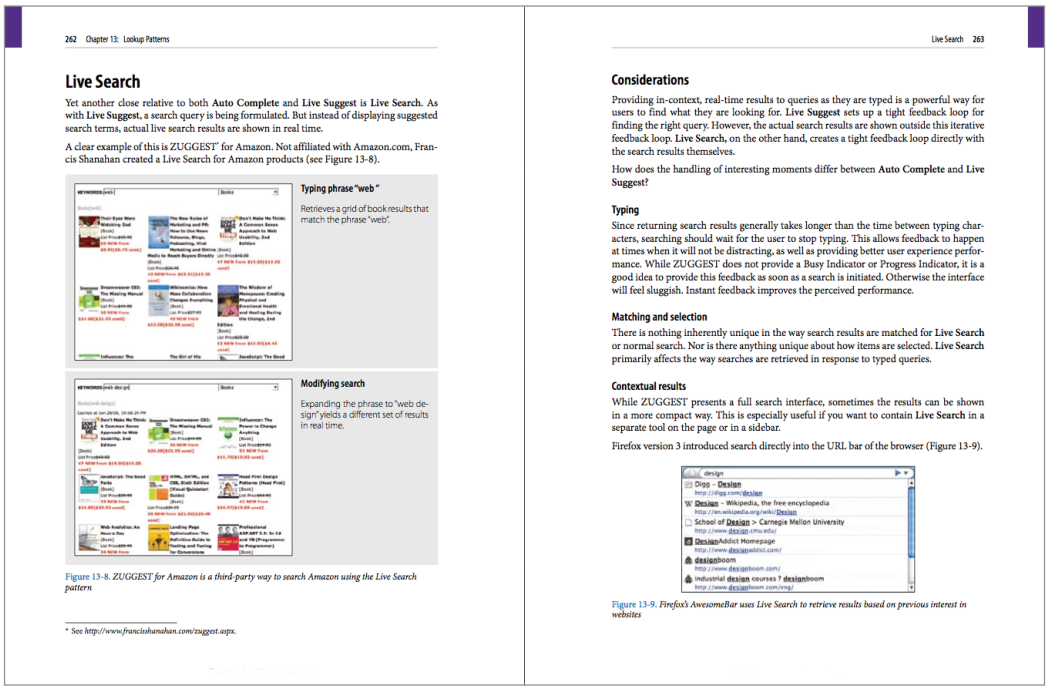
Published in 2009, *Designing Web Interfaces* is a book by Bill Scott and Theresa Neil with interaction design patterns for the web, most of them referring to the design of rich web interactions. These patterns handle, primarily, specific details of the user's interaction with the system, in contrast, for example with *The Design of Sites* (van Duyne, Landay and Hong 2006), which deals with a more structural level of websites design. There is also a supplementary website[29] with an outline of all the patterns, and updated examples.

---

27 www.patternry.com

28 In the original: "What problem does the pattern solve?", "When to use it?", "How to use it" and "Why to use it".

29 www.designingwebinterfaces.com



262 Chapter 13: Lookup Patterns

Live Search

Yet another close relative to both **Auto Complete** and **Live Suggest** is **Live Search**. As with **Live Suggest**, a search query is being formulated. But instead of displaying suggested search terms, actual live search results are shown in real time.

A clear example of this is ZUGGEST* for Amazon. Not affiliated with Amazon.com, Francis Shanahan created a Live Search for Amazon products (see Figure 13-8).

Typing phrase "web "

Retrieves a grid of book results that match the phrase "web".

Modifying search

Expanding the phrase to "web design" yields a different set of results in real time.

Figure 13-8. ZUGGEST for Amazon is a third-party way to search Amazon using the Live Search pattern

* See *http://www.francishanahan.com/zuggest.aspx*.

Live Search 263

Considerations

Providing in-context, real-time results to queries as they are typed is a powerful way for users to find what they are looking for. **Live Suggest** sets up a tight feedback loop for finding the right query. However, the actual search results are shown outside this iterative feedback loop. **Live Search**, on the other hand, creates a tight feedback loop directly with the search results themselves.

How does the handling of interesting moments differ between **Auto Complete** and **Live Suggest**?

Typing

Since returning search results generally takes longer than the time between typing characters, searching should wait for the user to stop typing. This allows feedback to happen at times when it will not be distracting, as well as providing better user experience performance. While ZUGGEST does not provide a Busy Indicator or Progress Indicator, it is a good idea to provide this feedback as soon as a search is initiated. Otherwise the interface will feel sluggish. Instant feedback improves the perceived performance.

Matching and selection

There is nothing inherently unique in the way search results are matched for **Live Search** or normal search. Nor is there anything unique about how items are selected. **Live Search** primarily affects the way searches are retrieved in response to typed queries.

Contextual results

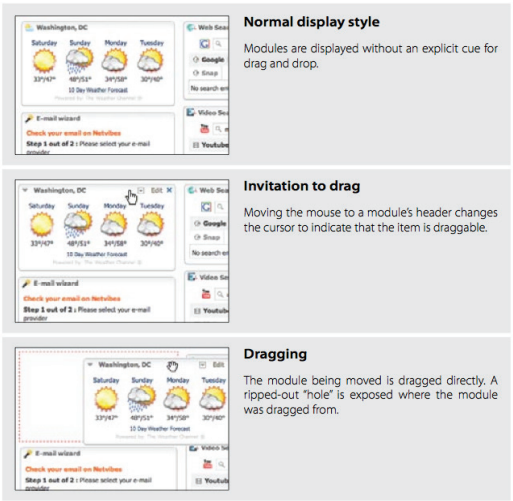
While ZUGGEST presents a full search interface, sometimes the results can be shown in a more compact way. This is especially useful if you want to contain **Live Search** in a separate tool on the page or in a sidebar.

Firefox version 3 introduced search directly into the URL bar of the browser (Figure 13-9).

Figure 13-9. Firefox's AwesomeBar uses Live Search to retrieve results based on previous interest in websites

Figure 4.22 Overview of the *Live Search* pattern (Scott and Neil 2009 262–263).

The library presents seventy-one patterns, structured on a three-level hierarchy with six major categories on the first level, arranged around what the authors call 'design principles': imperative sentences of basic rules that one must follow if they want to create a functional website, e.g., *Make it Direct* or *Keep It Lightweight*. The second level is arranged by patterns of similar function, e.g., the subsection *Direct Selection* collects patterns with different types of selections. Sections and subsections are numbered, but those numbers do not appear to reflect an order of importance between sections, they are probably used so patterns can be easily traced inside the book.



Normal display style

Modules are displayed without an explicit cue for drag and drop.

Invitation to drag

Moving the mouse to a module's header changes the cursor to indicate that the item is draggable.

Dragging

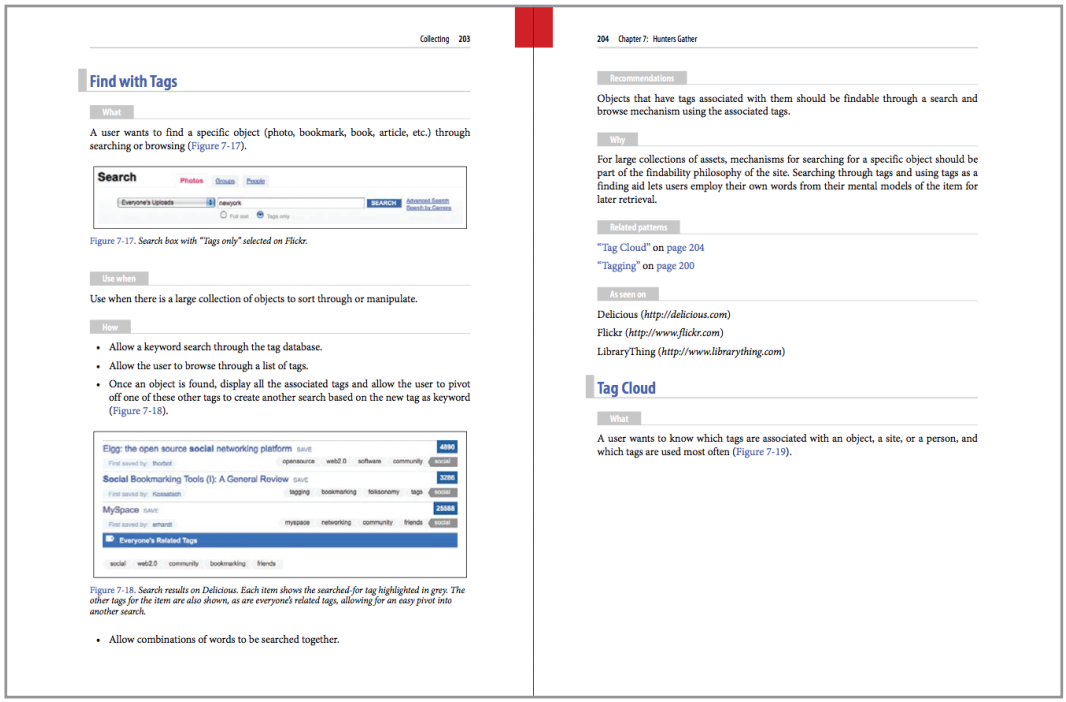
The module being moved is dragged directly. A ripped-out "hole" is exposed where the module was dragged from.

Figure 4.23 Illustration for the *Drag and Drop Module* pattern (Scott and Neil 2009, 30).

Patterns have a name (that is used throughout the book in bold), a small introduction, a main illustration, and *Considerations*. This is the longest part of a pattern and the place where the pattern explanation is given; this section varies considerably between patterns and can be complemented with additional images. In contrast with many of the other libraries, the solution — at least as it is perceived in Alexander's patterns — is not explicitly stated and must be inferred from reading the pattern. Most patterns have at the end, an additional section with the pattern best practices, where a series of recommendations are summed up on a list in a very direct and succinct way.

Many of the examples offered in the book are about interactions of the user with the system, and since this work was published as a book, authors had to find a way to represent the different states of a screen. They chose to show a series of screenshots — similar to a storyboard, e.g., Figure 4.23 — representing the key states of that interaction. To help the understanding of these images, each one is annotated with information describing what is happening or has happened in that particular state.

4.1.12 *Designing Social Interfaces*



Collecting 203

Find with Tags

What

A user wants to find a specific object (photo, bookmark, book, article, etc.) through searching or browsing (Figure 7-17).

Search Photos Groups People

Everyone's Uploads newyork SEARCH Advanced Search Search by Camera

Full text Tags only

Figure 7-17. *Search box with "Tags only" selected on Flickr.*

Use when

Use when there is a large collection of objects to sort through or manipulate.

How

- Allow a keyword search through the tag database.
- Allow the user to browse through a list of tags.
- Once an object is found, display all the associated tags and allow the user to pivot off one of these other tags to create another search based on the new tag as keyword (Figure 7-18).

Figg: the open source social networking platform SAVE 4890

First saved by: thorbot opensource web2.0 software community social

Social Bookmarking Tools (I): A General Review SAVE 3286

First saved by: Kassatsch tagging bookmarking folksonomy tags social

MySpace SAVE 25668

First saved by: erhardt myspace networking community friends social

Everyone's Related Tags

social web2.0 community bookmarking friends

Figure 7-18. *Search results on Delicious. Each item shows the searched-for tag highlighted in grey. The other tags for the item are also shown, as are everyone's related tags, allowing for an easy pivot into another search.*

- Allow combinations of words to be searched together.

204 Chapter 7: Hunters Gather

Recommendations

Objects that have tags associated with them should be findable through a search and browse mechanism using the associated tags.

Why

For large collections of assets, mechanisms for searching for a specific object should be part of the findability philosophy of the site. Searching through tags and using tags as a finding aid lets users employ their own words from their mental models of the item for later retrieval.

Related patterns

"Tag Cloud" on page 204

"Tagging" on page 200

As seen on

Delicious (*http://delicious.com*)

Flickr (*http://www.flickr.com*)

LibraryThing (*http://www.librarything.com*)

Tag Cloud

What

A user wants to know which tags are associated with an object, a site, or a person, and which tags are used most often (Figure 7-19).

Figure 4.24 Overview of the *Find with Tags* pattern (Crumlish and Malone 2009, 203–204).

Designing Social Interfaces is a book by Christian Crumlish and Erin Malone (2009) that collects patterns dealing with the design of social interactions. Most of them are related to the design

of websites, with a few from desktop applications. A supporting website[30] is also available, with almost the same content as the book but with some considerable differences in how the patterns are organized.

The book is divided in five chapters with ambiguous titles, which most of the time do not explicitly describe what they are about (e.g., *A Beautiful Day in the Neighborhood*), and with subchapters that are not much more explanatory (e.g., *Where's the Rest of Me?*). This approach to the naming of sections could be understood with the difficulty to summarize on a phrase all types of interactions that this book comprises, nevertheless, on the online version we can find a very similar structure but with more descriptive titles, e.g., the chapter *Object of My Desire* in the book, is named *Activities* in the online version. Chapters and sub chapters are numbered, but the numbering does not necessarily reflect a hierarchy.

Each pattern starts with a name, which may be followed by a succinct introduction with an image illustrating it; although sometimes the introduction and illustration may be absent. Patterns are organized in a minimum of four sections: *What*, *Use When*, *Why*, and *How*; depending on the pattern, other sections may be developed, such as *Special Cases* or *Consideration*. The extension and type of content can vary considerably between patterns, with the text formatted as prose or as a list.

In contrast with other libraries, there is not always a main image illustrating each pattern: some patterns may have multiple images, some none. A section dedicated to examples may exist, but these can be images as well as text. Despite the existence of this section, examples may appear in other sections, and each image is labeled with a description and a reference to the website or application from where the screenshot was taken.

Finally, patterns may have references to related patterns or a section, *As seen on*, with links to websites where these patterns can be found.

The content of the website is similar to that of the book, but not the same: it seems to have been rewritten for the book. Another difference between the two versions is the nomenclature used in the name of the pattern sections. What is named as *What*, *Use when*, *How* and *Why* in the book, in the website is, respectively, *Problem*, *Context*, *Solution*, and *Rationale*. Furthermore, even the text itself, though addressing the same points is written differently, and without the same level of development.

4.1.13 *UI Patterns*

UI Patterns[31] is a platform of a Danish web developer, Anders Toxboe, dedicated to user interface patterns and created with the purpose of filling some gaps of the *Yahoo! Design Pattern*

---

30 www.designingsocialinterfaces.com

31 www.ui-patterns.com

Library. It is a website with some emphasis on users' participation, where users are able to submit new patterns (although, they must first be approved by the site's owner), to add new examples (if they are registered), and to vote or comment. Registered users are also able to create and organize their own collections but, in spite of these features, at the date of writing, most patterns in the site were created by Toxboe.

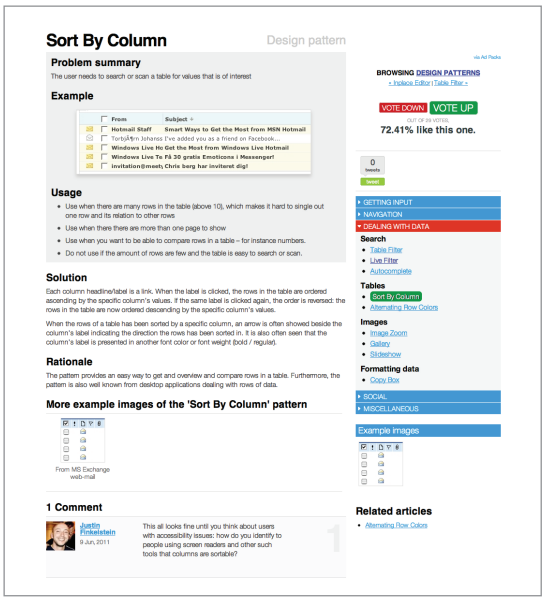


Figure 4.25 Overview of the *Accordion Menu* pattern (Toxboe 2007).

The site is organized in two distinct parts: one for design patterns, where we can find the pattern itself with the reasoning behind it; a second with multiple examples, submitted by users, from sites where the pattern is used. For each part there is a corresponding section on the other part, i.e., for the pattern *Breadcrumbs* there is a matching section with examples, and vice versa. Nonetheless, the example's page is linked on the pattern page but not the other way around — if we are looking for examples we cannot find a direct link to its reasoning.

In the part of the site with examples, we can browse by categories — corresponding to the patterns in the other part of the site, but here, placed at the same hierarchic level. Patterns can be filtered by tag, by color (the predominant colors of its graphics) or by domain (the sites' domain where we can find them). As well as patterns, each example can also be voted. The content of this part of the site is mostly sustained by users' participation through the uploading of new examples.

In the part of the site dedicated to its explanation, patterns are organized by categories in a three-level hierarchy. In the first level there are five main categories, related to the type of user's actions or objectives; and a *Miscellaneous* category. These categories mostly serve to organize related patterns, as it does not appear to exist an explicit order between them. Each one is also further divided by subcategories of related content.

On the pattern itself, we can find a name followed by a highlighted block with the problem summary, usually no more than a sentence; a main illustration, which can be a single image, or a series of images when the pattern deals with dynamic content; and finally, the *Usage*, which presents a list with the cases where we should or should not use this pattern. Next, follows the *Solution*, containing all the necessary steps for the implementation of the pattern; the *Rationale*, with the reasons of why we shall use this pattern; and occasionally, a *Discussion* section with other relevant considerations. In addition to the main example, additional ones may be offered; all of them are linked on the main page but dwell on the parallel part of the site dedicated to examples.

These patterns also have a rating, but with values less discrete than the ones used on Alexander's or Yahoo!'s patterns. Here, users can vote up or down on patterns and the result is shown in percentages. The term associated with the users' votes is *like*, e.g., "91% like this one"; however, the adjective used to describe the pattern might not be the most suitable, for example, classifying a pattern in terms of usefulness or efficacy would be more appropriated.

Depending on the pattern, related content may also be provided, such as: links to related patterns; websites that use it; or how it is represented in other libraries.

4.1.14 *Banco de Padrões de Design*

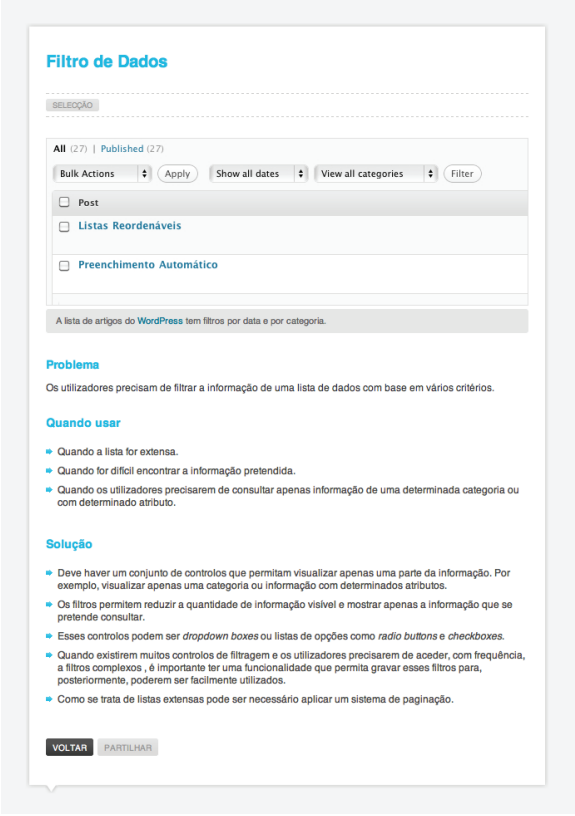


Figure 4.26 Overview of the *Filtro de Dados* (Data Filter) pattern (Instituto Superior Técnico 2010).

Developed by a team from *Instituto Superior Técnico*, it is the only reviewed library[32] that is not written in English. It provides a set of twenty-eight patterns, created with the purpose of empowering its academic community with tools to independently develop websites. Some patterns are based on those of the *Yahoo! Design Pattern Library*, as it is acknowledged.

Patterns are divided in four groups: *Forms*, *Interaction*, *Navigation*, and *Selection*;[33] with no hierarchy between the groups or the patterns within them. They are disposed with no apparent order, probably by creation date.

Patterns are composed with a minimum of four sections:[34] definition of the problem, when to use, solution, and examples — a major example, and occasionally, additional ones. Moreover, other sections or subsections may be developed, such as, sections related to the layout or accessibility.

Patterns reveal different levels of development, and it is difficult to recognize if they are unfinished or incomplete; we can find patterns with not much more information than a sentence by section, and others with multiple subsections and images, and almost all of its texts are written as lists.

The examples provided have a label describing it, with a link to the website from where the screenshot was taken. However, since those links take us to the homepage and not the page with the example itself, they are not always very helpful. And since no timestamp is available, we cannot know if the example is still active. One of the patterns (*Cursor Invitation*) also has a small video displaying its functioning, but other patterns, especially the ones from the same group (*Interaction*), could be improved with this kind of examples.

4.1.15 *Designing Mobile Interfaces*

Designing Mobile Interfaces is a book by Steven Hoover and Eric Berkman published in 2011 and dedicated to design patterns for mobile devices. These patterns cover multiple types of devices, whether they are smartphones, feature phones, or another type of mobile devices (e.g., portable media players). The book comprises patterns for the design of apps, websites or even the underlining operating systems. There is an additional website,[35] in a wiki format, with the same content as the book.

There are in total seventy-six patterns spread across four sections: *Page*, *Components*, *Widget* and *Input and Output*; each one further divided in subsections. Sections appear to have been sorted in terms of scale: starting with larger elements, until the more shapeless ones, e.g.,

---

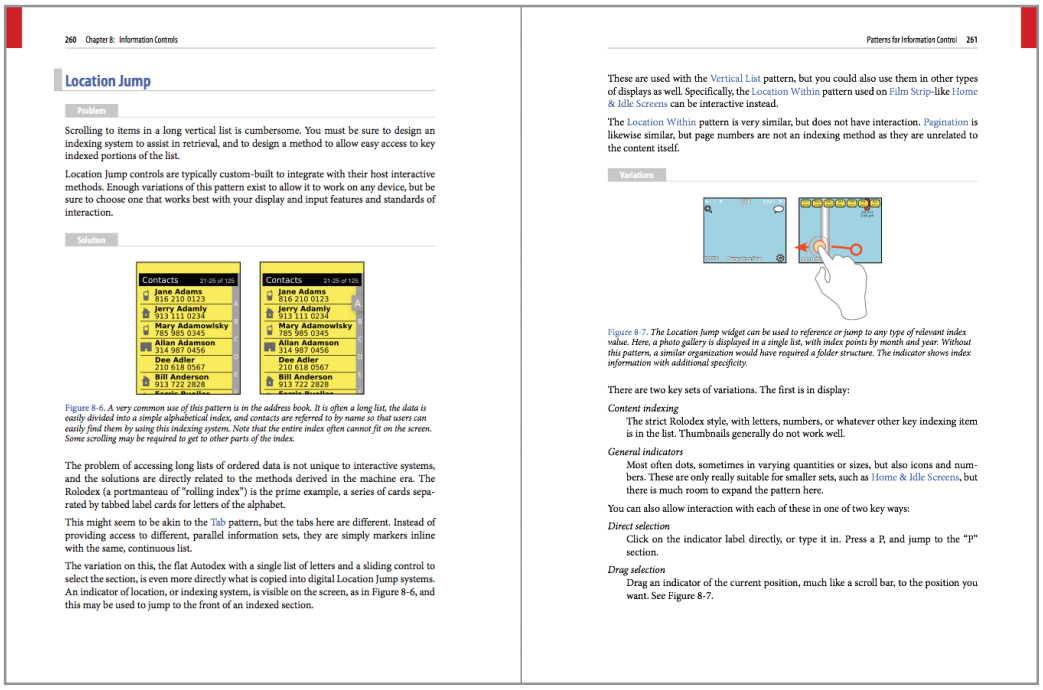
32 www.bpd.ist.utl.pt

33 In the original, respectively: *Formulários*, *Interacção*, *Navegação* e *Selecção*.

34 In the original: *Problema*, *Quando Usar*, *Solução*, and *Outros Exemplos*

35 www.4ourth.com

Audio and Vibration. Patterns are also divided into thirteen smaller subsections, in terms of the pattern function. In contrast with the top-level sections that were sorted by scale, it is not clear whether there is an order in the subsections.



260 Chapter 8: Information Controls

Location Jump

Problem

Scrolling to items in a long vertical list is cumbersome. You must be sure to design an indexing system to assist in retrieval, and to design a method to allow easy access to key indexed portions of the list.

Location Jump controls are typically custom-built to integrate with their host interactive methods. Enough variations of this pattern exist to allow it to work on any device, but be sure to choose one that works best with your display and input features and standards of interaction.

Solution

Contacts 21-25 of 125
Jane Adams 916 210 0123
Jerry Adamly 913 111 0234
Mary Adamowisky 785 985 0345
Allan Adamson 314 987 0456
Dee Adler 210 618 0567
Bill Anderson 913 722 2828

Figure 8-6. *A very common use of this pattern is in the address book. It is often a long list, the data is easily divided into a simple alphabetical index, and contacts are referred to by name so that users can easily find them by using this indexing system. Note that the entire index often cannot fit on the screen. Some scrolling may be required to get to other parts of the index.*

The problem of accessing long lists of ordered data is not unique to interactive systems, and the solutions are directly related to the methods derived in the machine era. The Rolodex (a portmanteau of "rolling index") is the prime example, a series of cards separated by tabbed label cards for letters of the alphabet.

This might seem to be akin to the Tab pattern, but the tabs here are different. Instead of providing access to different, parallel information sets, they are simply markers inline with the same, continuous list.

The variation on this, the flat Autodex with a single list of letters and a sliding control to select the section, is even more directly what is copied into digital Location Jump systems. An indicator of location, or indexing system, is visible on the screen, as in Figure 8-6, and this may be used to jump to the front of an indexed section.

Patterns for Information Control 261

These are used with the Vertical List pattern, but you could also use them in other types of displays as well. Specifically, the Location Within pattern used on Film Strip-like Home & Idle Screens can be interactive instead.

The Location Within pattern is very similar, but does not have interaction. Pagination is likewise similar, but page numbers are not an indexing method as they are unrelated to the content itself.

Variations

Figure 8-7. *The Location Jump widget can be used to reference or jump to any type of relevant index value. Here, a photo gallery is displayed in a single list, with index points by month and year. Without this pattern, a similar organization would have required a folder structure. The indicator shows index information with additional specificity.*

There are two key sets of variations. The first is in display:

Content indexing
The strict Rolodex style, with letters, numbers, or whatever other key indexing item is in the list. Thumbnails generally do not work well.

General indicators
Most often dots, sometimes in varying quantities or sizes, but also icons and numbers. These are only really suitable for smaller sets, such as Home & Idle Screens, but there is much room to expand the pattern here.

You can also allow interaction with each of these in one of two key ways:

Direct selection
Click on the indicator label directly, or type it in. Press a P, and jump to the "P" section.

Drag selection
Drag an indicator of the current position, much like a scroll bar, to the position you want. See Figure 8-7.

Figure 4.27 Overview of the *Location Jump* pattern (Hoober and Berkman 2011, 260–261).

Patterns are identified by a name that is styled throughout the book in color and capitalized. This is followed by a section describing the problem; and another section describing the solution that is accompanied by an illustration.

Authors chose not to include screenshots of real interfaces to illustrate each pattern, instead, they opted to use a low-fidelity image, similar to a wireframe, with a schematic and abstract representation of the pattern. They produced illustrations that shared a common language, for example, when color is used it has some meaning: interactive elements are styled in yellow; blue is used for images or graphics; gray represents non-selectable items; and orange is used when the item is in focus, e.g., when scrolling a list. They considered that screenshots might not display clearly the pattern, since each device adds its own style, and these additional graphic elements could overshadow the pattern itself. Additionally, they also chose to use illustrations for a reason of practicality: some of the examples needed were from devices where it was difficult to take clear pictures, e.g., feature phones or GPS.

At the end of the pattern there is a section with possible variations of the pattern, e.g., the *Vertical List* pattern suggests as an alternative, the pattern *Infinite List*.

The section *Interaction Details* explains the interaction of the user with the pattern, such as, places where the user can click and what happens when they do it. Similar to the previous

section, there is the section *Presentation Details*, which discusses the elements of the pattern that cannot be interacted by users. Both sections may be complemented with illustrations of the interaction, and graphics of hands interacting with the elements of the patterns are usually used to help in the understanding of the pattern.

At last, the section *Antipatterns*, discusses the situations when we should not use this pattern, or problems that may arise with its implementation.

4.1.16 *Mobile Design Pattern Gallery*

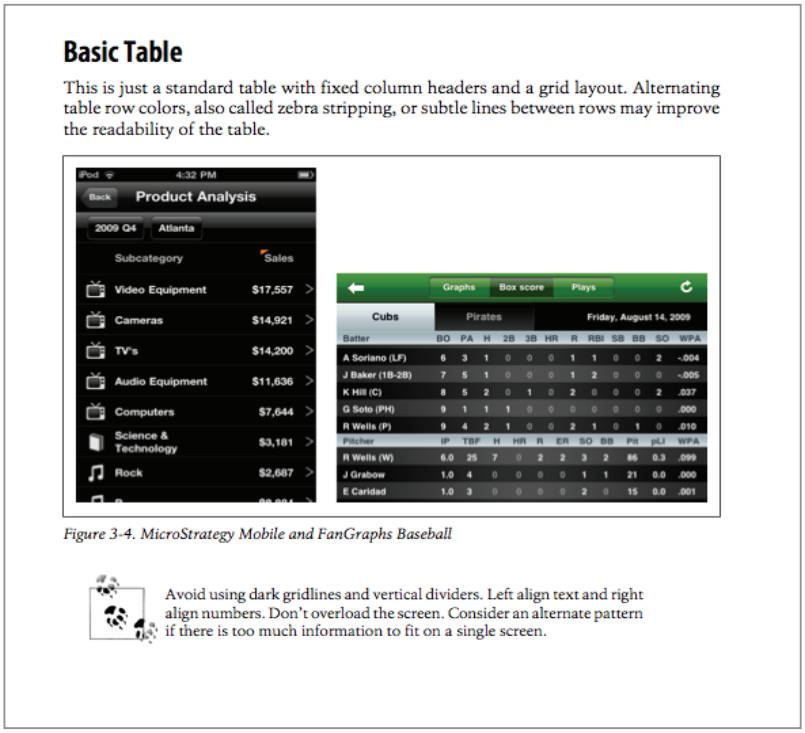


Figure 4.28 Overview of the *Basic Table* pattern (Neil 2012, 68).

This in one of the latest pattern libraries published and one of the few that only contains patterns for mobile devices. This library — or gallery, as they appropriately describe it — is a two-part work: a book and a website.[36] It is composed by patterns for the design of mobile applications, with examples collected from multiples devices and platforms. There are no examples from websites, but nevertheless, some are generic enough to be used on the web.

The patterns presented are somehow incomplete, at least if one follows Alexander's model. Despite being described as patterns, they are at best very simplified versions: they are composed by a name; screenshots from interfaces that use the pattern; a short description; and a sketchy illustration (Figure 4.29), in which the pattern is abstracted to a simplified and stylized form.

36 www.mobiledesignpatterngallery.com

Figure 4.29 Illustration for the *Expanding List* pattern (Neil 2012, 240).

Although not as complete as other libraries, this kind of work can be very helpful if we want a general and quick overview of how a pattern looks, or a usable source of inspiration. The usefulness of this work might also be augmented if we can link each pattern to a matching pattern in other libraries.

4.2 Summary of the Results

The work of Alexander has been providing a common ground for what is a pattern language; however, that does not mean that the pattern format he proposed has been followed faithfully.[37] The departure from the strict format of Alexander's patterns may be understood as a need to reach a broader public, but also as the result of the experience gained by authors through the years while working with patterns. An example of this can be found in Tidwell, in the way how her earlier work is closer to the format of *A Pattern Language* than the later.

A Pattern Language (Alexander et al. 1977) established a set of typographical rules that make the pattern structure visible without being explicitly declared, stipulating a clean and effective outline for patterns without cluttering the design with labels. However, this format did not get much support by other authors. While effective, implicit sections with long paragraphs of text can be rather unintelligible for someone who is trying to read a pattern for the first time. Thus, authors departed from this model towards one more direct and comprehensible, through the use of more and smaller sections, and more descriptive names, allowing an easier scanning of the content.

---

37 Only Borchers followed strictly the format of *A Pattern Language*, but *Common Ground* (Tidwell 1999), and *The Design of Sites* (van Duyne, Landay and Hong 2006) have a somehow comparable structure in terms of sections and content.

Alexander defends that patterns should be written in prose. Borchers also asserts that patterns should be presented as written texts "to make them easy to read and understand even for people from other professions" (Borchers 2001, 54), and avoiding "genre-specific jargon and notation where possible" (43). Regardless of the idea, this writing style has been dropped for something more direct and concise. The works of Alexander and Borchers have a rather academic tone, written in a quite technical language that may be a little intimidating for a reader trying to quickly decode the pattern purpose. For example, books published by O'Reilly (e.g., *Designing Interfaces*) aim to reach a broader public, therefore, they are written in a friendlier style, with fewer and shorter paragraphs, and with a widespread use of lists, combined with several images. Online libraries also follow this more direct language.

Alexander carefully ordered his language to reflect the spatial relations among patterns, from the ones of larger scale to the building details. This organization worked particularly well because architecture deals predominantly with physical relations. Although this order is not so clear in patterns for interaction design, authors have tried to propose systems that respond to this problem. For instance, van Duyne proposed an order in which "the earlier the pattern group appears on this scheme, the earlier it should be used in the design process" (2006, 31). Tidwell uses a similar system, in which she sorted patterns "by their approximate order in the design progression" (Tidwell 2011, xx) that, in short, is translated as: first, it is given the users' behavior; then, the structural components; and finally, the details.

Patterns libraries are generally organized in some sort of hierarchical system that provides a more manageable list, helpful when one needs to find a particular pattern. Only libraries with few patterns do not have a rigid hierarchic system in place, but even in those it is useful to have some sort of organization; for example, *Patternry* uses a tag system for that purpose. From the set of libraries analyzed, thirteen use a two- or three-level scale hierarchy, and one (*Yahoo! Design Pattern Library*) has a hierarchy with four levels.

4.2.1 Name

Naturally, every pattern needs a name, so it can be called and recalled during a design project, which needs to be strong and clear so clearly communicates its purpose. Alexander considers that "the search for a name is a fundamental part of the process of inventing or discovering a pattern" (1979, 267). Tidwell (1999) recommends the avoidance of "GUI-centric" names whenever possible (e.g. mice, menus, dialogs), so patterns can be used outside the domain of graphic user interfaces. A good example is expressed in *The Design of Sites* (van Duyne, Landay and Hong 2006), in which the name is designed as a phrase so it could be used within a sentence, e.g., "What is the name of that *Page Template*?". However, in other libraries the nomenclature of patterns is not so consistent. For example, in *Designing Social Interfaces* (Crumlish and Malone

2009) we can find in the same library patterns named as nouns (*Forums*), actions (*Saving*), or phrases: some imperative (*Edit This Page*), some metaphorical (*The Ex-Boyfriend Anti-pattern*).

When a pattern is referenced in another section of the library, it is a common practice to visually emphasize it. However, how this should be done is left to the author or publisher. Since Alexander started to compose the pattern name in small caps, some authors followed him, e.g., *A Pattern Approach to Interaction Design* (Borchers 2001), but that does not mean that this is the only, nor the best, way to format it. In the books from O'Reilly, e.g., *Designing Interfaces* (Tidwell 2011), the name is capitalized and in color, while in the online libraries, the name is, as expected, linked to the corresponding pattern, and styled with a contrasting color or underlined.

In some libraries, a numeric or an alphanumeric code is used to identify patterns. Alexander numbered his patterns sequentially from 1 to 253, reflecting the hierarchy between them. A more interesting approach is used, for example, in *The Design of Sites*, in which a unique alphanumeric code identifies the pattern and the hierarchic structure of the language (e.g., *F3 Shopping Cart*, the 'F' refers to the group and the '3' to the position of pattern in the group). *Patterns for Computer-Mediated Interaction* (Schümmer and Lukosch 2007) uses a similar system, with the difference that, there is not a dedicated index for the patterns, so the index of the book chapters is used to this purpose (e.g., *3.1.6 User Gallery*). These type of codes are more useful in print works, since books are not inherently open to constant changes. Indeed, none of the online libraries use this type of system. For example, in libraries such as *UI-Patterns* (Toxboe 2007) it would not be advisable to number patterns — other than by the order that they were added to the library —, since at any time a new pattern can be added to the library, and thus, break the relations established previously. In *The Form of Facts and Figures* (Behrens 2008a) each name is accompanied by a unique pictogram, with a very simplified version of the pattern solution (Figure 4.30). Although helpful and a nice touch, given the scale of those images, it might be difficult to extend this strategy to other domains.

Figure 4.30 *Tree Diagram* pattern from *The Form of Facts and Figures* (Behrens 2008a, 98).

4.2.2 Ranking

Most libraries do not classify patterns, and in those that rate them, the most common type of ranking is a three-level scale, probably grounded on the one originally used by Alexander, corresponding to the authors' confidence in a pattern. However, only Borchers and Schümmer

use the same ranking as Alexander, i.e., the same asterisks with the same meaning. The *Yahoo! Design Pattern Library* uses a similar three-level scale, but it is expressed as textual expressions: *Beta*; *Working Solution*; *Best Practice*.

In two of the online libraries, patterns are classified by the users' input; in *UI-Patterns* and *Info Design Patterns* users can vote up or down for a pattern. The main difference between the two is that one presents the result in absolute values, for example: 8 persons 'approved' the pattern (*Tree Diagram*, in *Info Design Patterns*); and the other as a percentage (Figure 4.31).

Figure 4.31 *Article List* rating from *UI Patterns* (Toxboe 2007).

4.2.3 Context

In *A Pattern Language* (Alexander et al. 1977), each pattern is introduced by a small paragraph contextualizing the pattern within the language through the reference of other patterns; however, this information does not assume the same relevance in other libraries. Some have a short introduction for each pattern, but these do not have the same characteristics of the context, i.e., higher level patterns are not reference as a way to contextualize that pattern in the language. One reason for this may be that patterns in Alexander's book function as a whole, and that the relations and even the order where they appear in the language have considerable significance, whereas, other authors do not give the same emphasis to the language as he gives.

4.2.4 Illustration

All libraries provide some sort of illustration for patterns, with the exception of *Common Ground* (Tidwell 1999) that only lists the examples textually. The type of illustration differs depending on the domain of the pattern and the medium of the library. In architecture, Alexander uses photographs of buildings and spaces. Borchers uses a diverse set of examples, the most common being photographs — mainly for patterns that represent people interacting with a system —, but also musical notations, organizational charts, and screenshots. In *Patterns for Computer-Mediated Interactions*, authors opted to illustrate patterns with symbolic photographs, usually a literal interpretation of the pattern's name, e.g., the pattern *Letter of Recommendation* is a photo of someone writing a letter; whereas patterns for interaction design use, predominantly, screenshots of the interface.

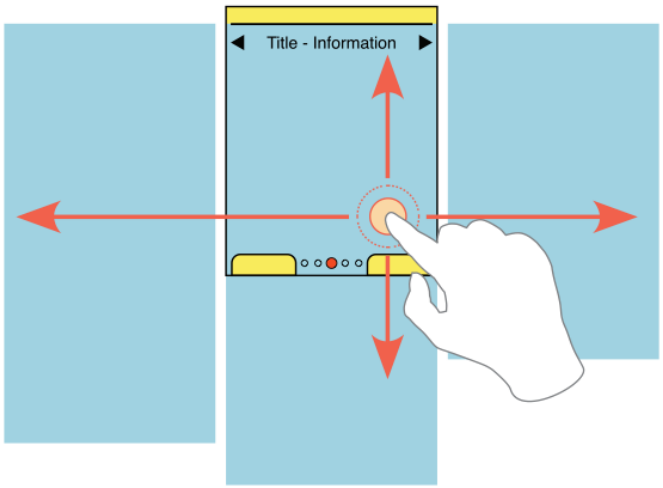


Figure 4.32 Illustration for the *Film Strip* pattern, in *Designing Mobile Interfaces* (Hoober and Berkman 2011, 91).

One important aspect of the interaction design patterns, is of the time dimension, i.e., the representation of the users' actions with the system. It is commonly denoted by the use of multiple images, each one portraying a different state of the system, and working analogously to a storyboard. The best example is the book *Designing Web Interfaces* (Scott and Neil 2009), in which almost all patterns are illustrated by a series of annotated screenshots, corresponding to the pattern's solution.

In *Designing Mobile Interfaces* (Hoober and Berkman 2011), patterns that denote an interaction between the user and the interface have graphics representing those gestures, e.g., the graphic of a hand with arrows embodying the movement needed to interact with the system (Figure 4.32).

An alternative approach to this problem is the use of small videos showing the user interacting with the system. However, this method is not used extensively, we only found a few examples in *Banco de Padrões de Design* (Instituto Superior Técnico 2010) and *Info Design Patterns* (Behrens 2008b).

Figure 4.33 Illustration for the *Checkbox* pattern, in *Info Design Patterns* (Behrens 2008b). The checkboxes on the top right can be ticked to toggle the visibility of the map layers.

The most interesting and noteworthy approach was used by Behrens (2008b) in *Info Design Patterns* through the application of real interactions, rather than merely static illustrations, depicting how a pattern really works. For example, in the pattern *Checkbox* (Figure 4.33) we can actually use the checkbox to toggle the visibility of the layers on the image. This approach allows us to apprehend more easily how interface elements respond to users' actions or how the system responds through time.

Besides the main example, it is common to have additional ones, either on a dedicated section (*Designing Mobile Interfaces*), throughout the pattern itself when needed (*Designing Interfaces*), or even through a whole website section dedicated to examples (*UI Patterns*). In some online libraries examples are complemented with users' content.

4.2.5 Problem

Alexander summarizes the pattern problem in a paragraph styled in bold and spatially detached from the rest of the text. This format was initially followed in *A Pattern Approach to Interaction Design* (Borchers 2001) and in *The Design of Sites* (van Duyne, Landay and Hong 2006), but dropped by most authors since then. The change has been towards a more explicit label for the section. For instance, the problem may be presented as *What*, or even more descriptive as "What Problem Does This Solve?". Nonetheless, regardless of the nomenclature used, a statement describing the problem that the pattern addresses is somehow present in all patterns.

4.2.6 Forces

In *A Pattern Language* (Alexander et al. 1977) the reasoning for the pattern appears in a section between the problem and the solution statements. This is usually the longest part of a pattern, and in the case of Alexander and Borchers it is written in a dense block of text. To make this content more approachable, most authors opted, since then, to divide it in multiple and smaller sections, with a more direct writing style and broader use of lists.

The meaning of the expression *forces* might not be entirely clear for readers, thus, most authors omitted the reference to it. Indeed, the section that states the main reasoning for the pattern is only named as *Forces* in *Common Ground* (Tidwell 1999) and in *The Design of Sites (van Duyne, Landay and Hong 2006)*. Instead, the corresponding content is normally disposed in more descriptive sections, such as, *Why* or *Considerations*.

4.2.7 Solution

Alexander clearly states the solution of the pattern in a succinct paragraph visually detached from the rest of the text, written in a prescriptive language. Since all patterns need to provide

a clear solution, this format was somehow followed by most authors, either through a section named *Solution*, or in an equivalent section.

The most distinct case is *Designing Web Interfaces*, in which the solution is not explicitly given, but can be understood by the reading of the pattern. Indeed, the main steps necessary to the pattern's implementation are revealed in the storyboard.

Patternry and *UI-Patterns* present the solution at the beginning of the pattern, in a highlighted block, containing a short description and illustration of the solution. By clearly setting the purpose of the pattern and presenting the solution, this approach helps the reader to get a quick overview without the need of further reading.

4.2.8 Diagram

Although particularly important in Alexander's work, the diagram is not present in many of the libraries later developed. As he defended: "If you can't draw a diagram of it, it isn't a pattern." (Alexander 1979, 267). This is especially true in architecture because it deals with physical elements and spatial relations, but in other domains the drawing of a diagram can be a more difficult task. Nonetheless, even the patterns in *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma et al. 1995), which do not have a concrete shape, have some type of graphic representation, namely, a diagram based on the object-modeling technique (Figure 4.34).

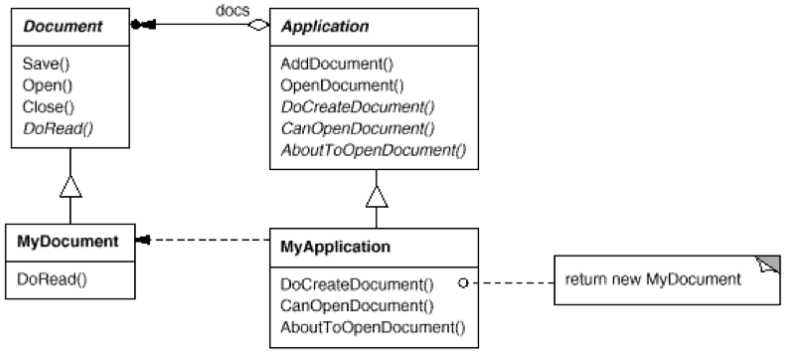


Figure 4.34 Diagram for the *Template Method* pattern, in *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma et al. 1995, 325).

With regard to interaction design patterns, few libraries provide a diagram. In *Common Ground* (1999), Tidwell occasionally uses drawings to represent a pattern, though, these sketches work primarily as illustrations, as they are used interchangeably with screenshots. The patterns in Borchers's work and in *The Design of Sites* (van Duyne, Landay and Hong 2006) are the only ones that have the solutions illustrated by diagrams (Figure 4.35); in Borchers'

work, the sketches assume different graphic styles depending of the domain of the pattern, e.g., musical notations for the musical pattern language.

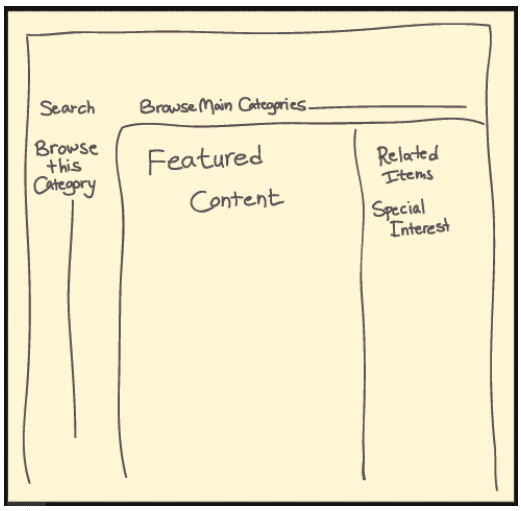


Figure 4.35 Diagram for the *Category Pages* pattern, in *The Design of Sites* (van Duyne, Landay and Hong 2006, 250).

4.2.9 Related Patterns

It is common for authors to establish some type of connection among the pattern in the language. While the nomenclature may differ, it is usually done through a dedicated section at the end of the pattern, referencing patterns that may complement that one. This information can be just a link to the pattern, but also an explanation of the connection. In other libraries, such as *Designing Mobile Interfaces* (Hoober and Berkman 2011), there is not a dedicated section for this; instead, the relations are explained within the text when necessary. Normally, the list of the related patterns is curated by the author, but can also be done programmatically by some criteria. For example, in *Patternry* all patterns have a list of five *Possibly Related Patterns* that are chosen based on common tags.

Generally these connections are established by a simple reference of the pattern's name, but sometimes the relation is explained. However, Behrens in *Info Design Patterns* (2008b), devise a more complex and interesting system, in which a pattern can be connected to another by one of five types of relations.

Although they are not the same kind of connections, some libraries acknowledge the existence of other works by linking to a corresponding pattern in another library (e.g., *UI-Patterns*). This information can be helpful given that it provides an alternative take in the same subject, but more important, it can complement and enrich the value of a pattern with additional material.

4.2.10 Comments

Online libraries, with the exception of *Banco de Padrões de Design* and *Info Design Pattern,* take advantage of the web to collect users' feedback. This is usually done through a section dedicated to comments, but can also be done through a forum, such as in *Yahoo! Design Pattern Library*. Despite having a laudable goal, the users' participation does not appear to be very significant or relevant. Many comments are from people praising the utility of the pattern, thanking the author, or simply meaningless dialogues rather than actual contributions or constructive critiques.

4.2.11 Code Examples

Three of the libraries provide code examples, though, each one with different functions and characteristics. Because of the domain, in *Design Patterns: Elements of Reusable Object-Oriented Software* (Gamma et al. 1995), code examples are not a secondary part of the patterns but rather one of the main components. *Yahoo!* provides a link to its developer network, with the technical information necessary for the implementation of the pattern. This information is particularly useful in the case of *Yahoo!* since the library was developed as an internal tool. *Patternry* has an optional section dedicated to code snippets, with content that can be added by users.

Depending on the patterns' domain, it may be helpful to have this kind of content available, especially if authors want to have a more pragmatic approach their work. Nonetheless, this kind of content is rather ephemeral and it might have to be regularly updated, so it is perhaps better if this section is not part of the main body of the pattern.

5 PATTERN MODEL

The comparative analysis conducted in the beginning of this project and described on the previous chapter allowed us to defend more assertively the template that is used to format the patterns in this work. Nevertheless, we end up with a format that resembles the one used in Alexander's patterns, but with some differences. In short, each pattern includes in the following order: *name*, *illustration*, *problem*, *solution*, *rationale*, *examples*, and *related patterns*.

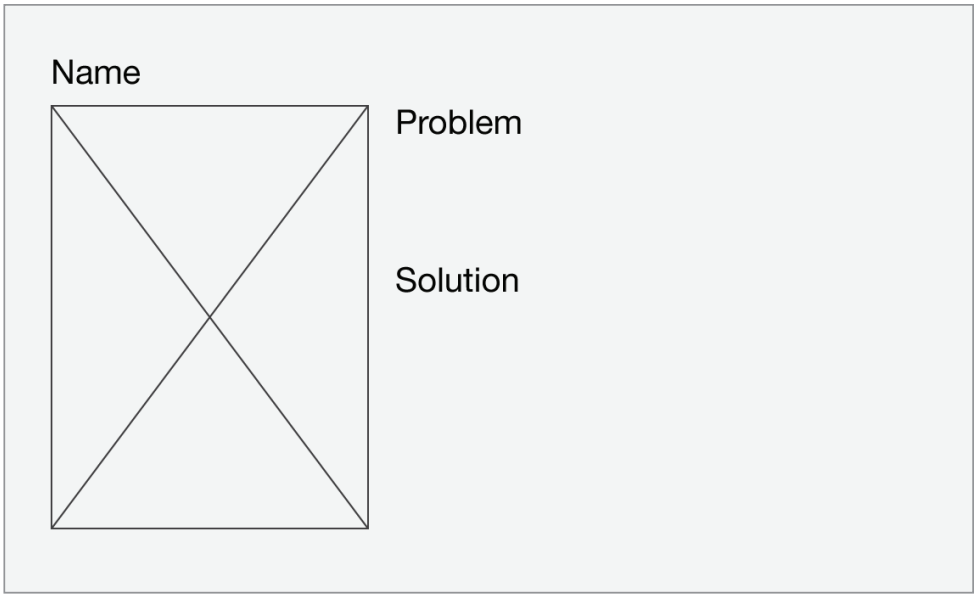


Figure 5.1 Scheme for the highlighted block that is used in the online version.

The first group of elements that form a pattern is composed by the *name*, *problem*, *solution*, and *illustration*, and work as a summary of the pattern. We devise the pattern model in a way that the reading of this first block provides an overview of the pattern, meaningful enough to introduce its general purpose. Therefore, it is recommended that one should start the reading of a pattern by these elements. On the print version, this block is separated from the rest of the pattern content by a series of three asterisks, while on the online version it follows the layout of Figure 5.1.

The problem and the solution statements work as a whole, and were written to be read in sequence, and posed in logical terms as: if *problem*, then *solution*. That is, if one identifies the problem stated as one of its own, it should apply the corresponding solution.

We nevertheless do no endorse the idea that the reading of the first block is enough for a successful implementation of a pattern. Although we have tried to provide the essential information in the problem and solution statements, alongside with the illustration, it is advisable to read the reasoning behind it, since most considerations that arise with the use of a pattern are not contemplated here.

Furthermore, additional considerations about the technical implementation of the pattern may be given. However, these notes are not part of the main body of the pattern, but appear as footnotes, since they address concerns with the implementation of the pattern in current devices and browsers that may be resolved in future updates, and thus do not inhibit the pattern itself.

5.1 Organization

Patterns are organized with an implicit hierarchy that can be inferred by the order in which they appear in this work. They are sorted in terms of scale, from the ones that address the macro structure of a design to ones related to interaction details, following a logical order that reflects the general process that one follows when designing a website. There are different groups of patterns related to its functions and characteristics, though they are not explicitly declared:

- The first group — LINEARIZED LAYOUT and GRID LAYOUT — contains two patterns that set up the foundation to the site layout, which its use determines the rest of a design;
- The second group — LINEARIZED MENU, JUMP MENU, TOGGLE MENU, SIDE MENU and SELECT MENU — presents five patterns to help us address the problem that is designing a navigation for smaller screens;
- The third group — VERTICAL LIST, INFINITE LIST, THUMBNAIL LIST, EXPANDING LIST, FIXED CONTENT — is responsible for laying out the different possibilities that one has for presenting content;
- The fourth group contains three patterns — SLIDESHOW, TABS, DROPDOWN — that imply some sort of progressive disclosure;
- The fifth group contains two patterns — LINEARIZED TABLE, ABRIDGED TABLE — for designing tables;
- The sixth group proposes two patterns — CLEAR ENTRY, DYNAMIC FILTERING — to help on the design of web forms;
- Finally, the seventh and last group — TOUCH FRIENDLY TARGET, ICEBERG TIP — addresses the problem of designing interfaces that need to be effortlessly used by touch.

5.2 Name

Throughout the work in a design project we need to somehow identify the problem we are tackling, thus, every pattern must have a name. Alexander considers "the search for a name is a fundamental part of the process of inventing or discovering a pattern" (1979). We tried to choose a name that explains the problem that the pattern addresses, being something succinct, direct and that may be used normally during a project discussion.

The name chosen is a proposal of one that we think that best fits the solution for the design problem at hand. Nonetheless, since the development of the pattern itself is usually an iterative process, other names may be suitable, thus, when feasible, we acknowledge comparable patterns and their respective names.

When a pattern is referenced by others, either on the *Related Patterns* section or throughout the pattern itself— as well as in the rest of this dissertation — its name is formatted in small caps to emphasize it from the rest of the text, and to offer a reminder that it is a pattern from this library. Patterns from other authors are set in title case and italicized.

5.3 Illustration

A major difference between the format proposed in this work, and others used in other libraries that deal with interaction design is the effort given to the *Illustration* section. An important aspect in the design of illustrations for interaction design patterns is the representation of the users' actions with the system. An inefficient illustration may prevent users from fully understanding a pattern. In general, authors tried to overcome this problem through the use of multiple images representing the different states of a pattern, comparable to a storyboard; through annotated illustrations; or simply through an image that tries to capture the most distinct moment of a pattern. Indeed, those would be our options if we were only dealing with printed media. However, none of these methods fully satisfied us, because in those cases details of the interaction and information of what happens between states are lost. Therefore, we tried to respond to this problem through the use of interactive illustrations — an approach also used by Behrens (2008b) in some of his patterns. As such, we explored this idea through the development of a hypothetical and archetypal website[38] that is used to demonstrate the patterns in the language.

Each interactive illustration is a basic web page, similar to a wireframe or a diagram that contains the structural elements of a pattern, but more important, it contains the interactive elements that shape it. They capture the main components that intervene in the design solu-

38 www.patterns.jribeiro.org

tion, providing an outline of the pattern, and allow users to interact with the different elements, e.g., users can tap a button and see what that will trigger.

These illustrations follow a simplified and coherent graphic language — analogous to the ones used by Hoober and Berkman (2011), and Behrens (2008b) —, with the purpose of removing irrelevant artifacts that a screenshot of a real website may contain, and to make them easier to compare. We hope that these interactive illustrations become a valuable asset that provides to the reader a better and precise representation how a pattern works.

In order to exemplify how patterns respond to different screen sizes, we devised a system that allows us to toggle between view modes: portrait, landscape, and desktop view. The portrait and landscape modes are essentially a frame that limits the viewport to a resolution of 320x480 pixels, while the desktop mode is the website viewed without any frame constraints. The desktop mode it is particularly useful to visualize how patterns work on a responsive design, and essential to understand some patterns, for instance, the LINEARIZED LAYOUT is not that meaningful unless it can be compared to a conventional website. Although it is possible to activate the desktop view in all patterns, only a few patterns have a dedicated layout in this mode. The alternate mode views were intended to facilitate the visualization of a pattern in wider screens, thus, when the illustration is accessed by a mobile device these mode views are not available, so the layout fills the entire screen.

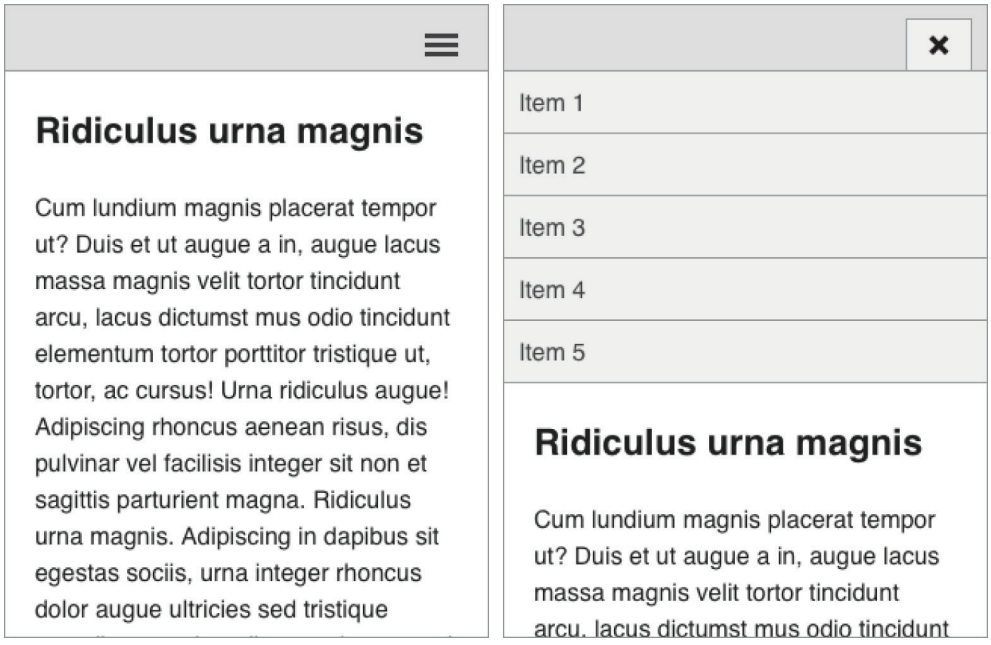


Figure 5.2 Illustration for the TOGGLE MENU, closed on the left, opened on the right.

Although we have dedicated some effort to the design and development of these interactive illustrations, we recognize that this work needs to be somehow self-contained. Therefore, on the print version, each pattern is also accompanied by an illustration that follows the same

language used on the online version. In general we used a single screenshot taken from the website, which tries to capture the moment that best synthesizes the design. However, for more complex scenarios, those that we considered that a single image was not enough to properly illustrate a pattern, we offer a series of images denoting each key state of the interaction. For example, in TOGGLE MENU — Figure 5.2 — we used two images, one for the closed state (left) and another for the open state (right).

Regardless of the effectiveness of these illustrations, real-world examples of interfaces that implement the pattern are not completely discarded, but rather listed in a dedicated section. Indeed, we consider that real examples an important part of a pattern that should contribute to its interpretation.

5.4 Problem

The problem is stated in succinct paragraph, briefly describing what is the purpose of a pattern, what problem it addresses, and when should it be used. It is posed as a hypothetical situation in which the problem occurs. Therefore, if one identifies the problem described as one of its own, it should proceed with a further exploration of the pattern.

5.5 Solution

Similarly to the problem, the solution statement is also presented in a concise paragraph, set as a generic instruction, written in a prescriptive language, and explaining how to solve problem described. We tried to be as brief as possible, but simultaneously present the fundamental principles that are in place in that pattern. However, for a more thorough explanation, the *Rationale* section should be consulted.

5.6 Rationale

We hope that the reader acquires the basic purpose of a pattern through the reading of the problem, the solution statement, and the corresponding illustration. Nevertheless, it is important to clarify and justify the options we made. Therefore, in this section we present a more extensive explanation for the pattern.

This is the longest section, so it may be divided in subsections for easier reading, and can contain any kind of relevant information: text, screenshots, graphics, or any other kind of appropriated content. It is the place where we explain the reasoning behind the pattern, proving a

theoretical background for it. We present the reasons of why one should use the pattern, when should and should not use it, as well as liabilities, problems, tips, concerns or implementations instructions that may be applicable.

5.7 Examples

Although a more abstract representation of a pattern can be superior when trying to grasp its basic structure, it is still important to see real-world examples. Therefore, at the end of the pattern, we present a collection of screenshots from websites that we consider a good example of the pattern's implementation, linked (when possible) to the page from where they were taken.

5.8 Related Patterns

Finally, at the end of the pattern there is a section with additional information that helps to expand and complement each one, consisting of two types of content:

- The first group consists of patterns in this library that have similar a function, and thus can be used as an alternative; or patterns with complementary functionalities.
- The second group is composed by equivalent patterns in other libraries. We acknowledge other libraries with patterns that accomplish similar functions, given that they can provide additional information for the same problem, without the need to repeat the same content. For example, the *Mobile Design Pattern Gallery* (Neil 2012) contains an extensive list of examples that can be a useful source of inspiration.

6 WEB DESIGN PATTERNS FOR MOBILE DEVICES

The research described in the previous chapters had as its main purpose the development and writing of design patterns. The above study contributed to the definition of the current state of art of the mobile media, particularly regarding the domain of web design; and provided us with a methodology for the writing of design patterns. Furthermore, to assist us in the writing of patterns, we developed three web design projects — which will be described in the next chapter — that allowed us to experiment with the design of mobile interfaces, as well as to validate the patterns that were being written. This research was later systematize and organized through the means of the patterns that are presented here.

In this chapter we describe a set of twenty-one patterns, developed with the objective of helping in the design of websites for mobile devices. The set of patterns proposed intends to be viewed as a work in progress, in the way that these patterns can be further developed, as well as new ones can be suggested. They address the problems of designing interfaces for the current devices and technology, hence they are receptive to improvements as the state of the mobile web progresses.

These patterns were created with the purpose to be used as a practical tool, one that could be easily consulted when needed throughout the course of a design project. To facilitate on this task, and so that this work would not become constricted to this dissertation, we published these patterns online,[39] with the hope to offer a more accessible medium for consulting them.

The idea, and sometimes the name, of the patterns included in this work came from two main sources: 1) other pattern libraries with comparable patterns, 2) from the research on designing websites for mobile; complemented with the work on designing mobile websites.

At the beginning of this project, we conducted a comparative analysis between libraries with patterns for interaction design, particularly those that already addressed the design of mobile interfaces. In this analysis we collected patterns that could be adopted and adapted to the design of mobile websites, and rewrote them to respond to the particularities that designing for the web on smaller screens impose. For example, the pattern VERTICAL LIST has

39 www.patterns.jribeiro.org

a comparable pattern in almost all libraries that were analyzed. It appears as *List Menu* in the *Mobile Design Pattern Gallery* (Neil 2012) or as *Vertical List* in *Designing Mobile Interfaces* (Hoober and Berkman 2011). Others came to life from reading and researching on the design of interfaces for mobile devices. For instance, the pattern ICEBERG TIP was inspired on the idea, and name presented in *Designing Gestural Interfaces* (Saffer 2008). Other patterns, such as TABS and DROPDOWN, which are quite common patterns on the web, were adapted in order to respond to the limitations that these devices require.

6.1 Linearized Layout

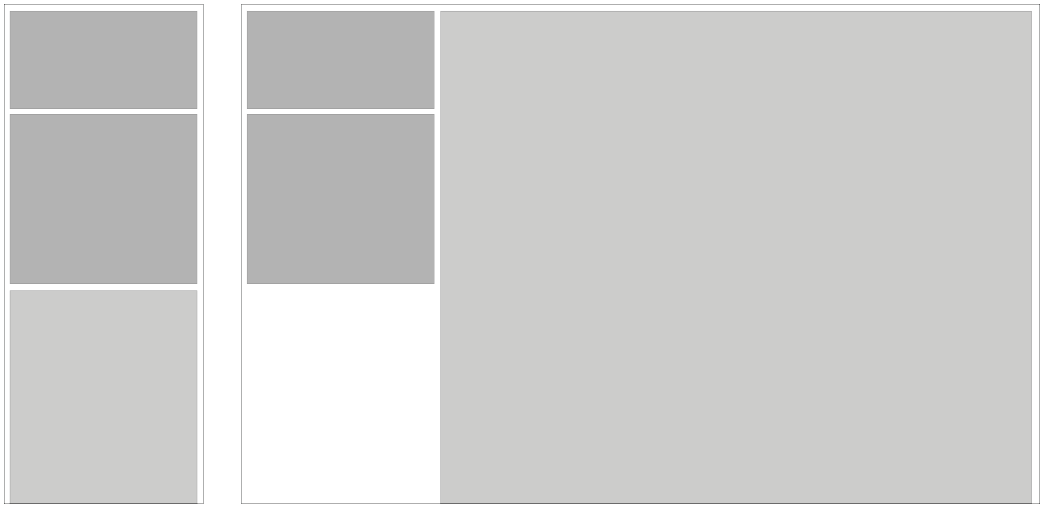


Figure 6.1 LINEARIZED LAYOUT <patterns.jribeiro.org/patterns/linearized-layout>.

Problem

On smaller devices with narrow screens, more complex designs with multiple columns do not fit perfectly. In most cases you end up with a scaled-down page that is unreadable.

Solution

Linearize the content, by stacking all blocks of information on top of each other, spanning the width of the screen.

***

Layouts with multiple columns are a common pattern on web design. However, they do not work properly on devices with narrow viewports. When this type of websites is accessed on these devices, browsers will zoom out the page until it fits on the width of the screen; or they will add horizontal scrolling. Both alternatives are far from satisfying. Column layouts should be optimized for mobile viewing. Therefore, you should design your mobile site by assembling the page vertically and by stacking each page section on top of each other, which is already the default behavior of a page when no style is defined.

Because we cannot foresee the size and resolution of the devices that will access our website, blocks with fixed widths are not a good practice. The LINEARIZED LAYOUT should have a fluid layout that adapts to the width of the browser whatever is its effective size.

This pattern is also a requirement for many of the other patterns proposed, particularly the VERTICAL LIST, i.e., if you want to use it you will invariably need to linearize the content.

An alternative approach the organization of content on a page is the GRID LAYOUT pattern.

***

Figure 6.2 *United Pixel Workers* <www.unitedpixelworkers.com>, April 2012.

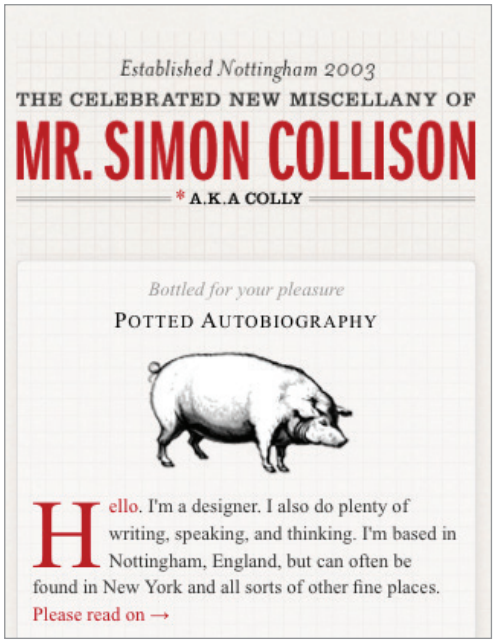


Figure 6.3 *Colly* <www.colly.com>, April 2012.

Related Patterns

- GRID LAYOUT
- VERTICAL LIST

- *Vertical Stack* in *Designing Interfaces* (Tidwell 2011)

6.2 Grid Layout

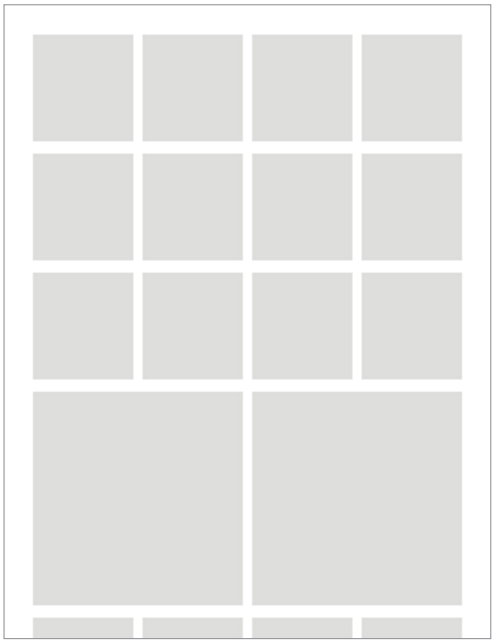


Figure 6.4 GRID LAYOUT <patterns.jribeiro.org/patterns/grid-layout>.

Problem

Although a LINEARIZED LAYOUT works in most cases, sometimes you have content that does not need to, or should not, fill the entire width of screen.

Solution

Arrange elements in a matrix of one or more rows.

***

While not as recurrent as the LINEARIZED LAYOUT, a GRID LAYOUT can be quite effective with some types of content. It is typically used with image content such as photos, illustrations or icons. The most common case is to present a gallery of images, but you can still use it with text, as long you keep it to a few words. Long paragraphs of text can become almost unreadable even if your grid only has two columns.

A GRID LAYOUT can have as many divisions as needed; however, if the grid blocks work as buttons they should be large enough and adequately spaced so they are easily triggered — use a TOUCH FRIENDLY TARGET or an ICEBERG TIP on those cases. Although in most examples blocks have the same height and width, they can still assume irregular forms. That is, blocks can span multiple columns or rows (Figure 6.5).

Figure 6.5 *Hillsong* website <www.hillsong.co.uk>.

Because with this pattern you are displaying multiple items side by side, in some cases it can be a more efficient use of the vertical space than a LINEARIZED LAYOUT. For example, if you have a VERTICAL LIST where each item is composed by only one word, you can save space by displaying several items on the same line (Figure 6.6).

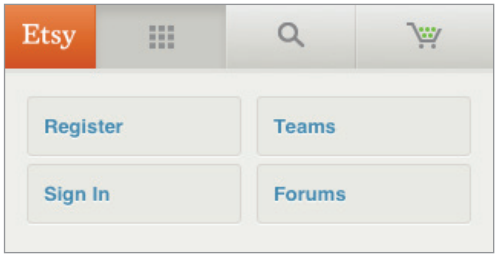


Figure 6.6 The TOGGLE MENU on *Etsy* website <www.etsy.com> reveals a menu that presents items on a grid.

Figure 6.7 The *Etsy* website <www.etsy.com> uses a SLIDESHOW that presents on the same slide multiple images on grid, each one working as a link.

The GRID LAYOUT can be used as a more structural pattern, or combined with other patterns to extend their capabilities. For example it can be used with a TOGGLE MENU to increase the number of items visible on the screen (Figure 6.6); or with a SLIDESHOW to increase the number of items by slide (Figure 6.7).

***

Figure 6.8 *dConstruct* <www.2012.dconstruct.org>, April 2012

Figure 6.9 *Jessica Hische* <www.jessicahische.is>, April 2012

Related Patterns

- LINEARIZED LAYOUT
- TOGGLE MENU
- SLIDESHOW

- *Grid* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)

6.3 Linearized Menu

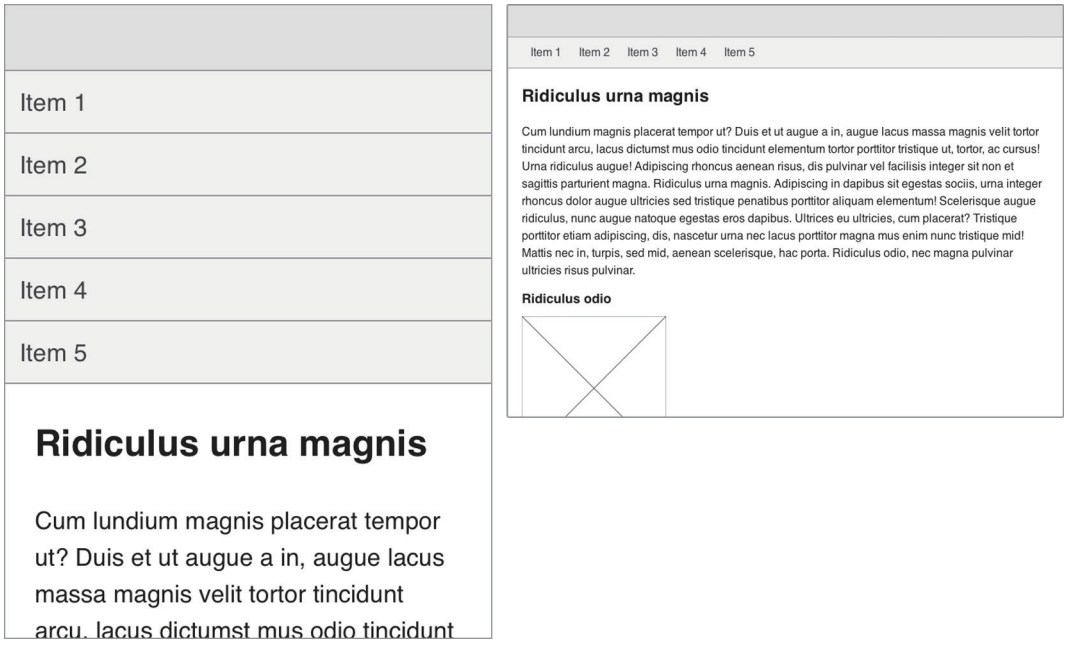


Figure 6.10 LINEARIZED MENU <patterns.jribeiro.org/linearized-layout>.

Problem

Although inline menus are quite common on desktop websites, they are difficult to achieve on mobile. If your menu has several items, it will not fit properly on the mobile version of the website.

Solution

List all menu items vertically spanning the width of the device.

***

Because of the narrow width of mobile devices, in most cases you do not have enough space to display items inline. The solution involves disposing list items vertically covering all the width of the screen. This type of menu is quite easy to implement because it is the default behavior of a list when not styled.

Be aware that if the menu is extensive and is positioned on top of the page it will probably fill most of the page. You can easily cope with this problem with a JUMP MENU.

You should optimize list items for touch by making the list span across the width of the screen, and tall enough so they are easily triggered and nicely spaced so the wrong target is not tapped by mistake — TOUCH FRIENDLY TARGET or ICEBERG TIP.

An alternative approach the organization of content on a page is the GRID LAYOUT pattern.

***

Figure 6.11 *New Adventures* <www.2012.newadventuresconf.com>, April 2012.

Figure 6.12 *Clean Air Works* <www.clearairchallenge.com>, April 2012.

Related Patterns

- VERTICAL LIST
- JUMP MENU
- TOGGLE MENU
- SIDE MENU

6.4 Jump Menu

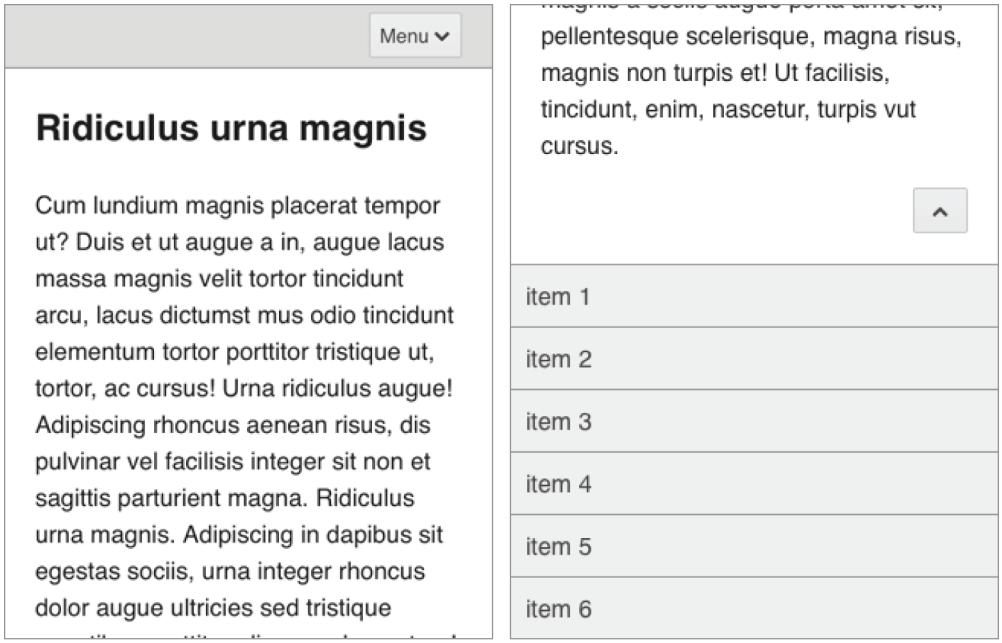


Figure 6.13 JUMP MENU <patterns.jribeiro.org/patterns/jump-menu>.

Problem

When placed on top of the page, an extensive LINEARIZE MENU will fill all the available space of the screen. However, generally, you do not want the menu to take precedence over the content.

Solution

Place the menu at the bottom of the page but display a button on top of the page that links to the menu.

***

A navigation menu with an extensive list of items can fill the entire screen when the page loads, relegating the content to second place. However, it is usually a good practice to emphasize content over navigation (Wroblewski 2011). This pattern tries to overcome this problem by focusing on the content while still providing quick access to the navigation. For that, you design a LINEARIZED MENU that is positioned at the bottom of the page while leaving a button on top that takes the user to the menu. Besides the button on top, it is helpful to provide a link next to the menu to take users back to the top, so they do not need to scroll the entire page if they need to go back.

Since the JUMP MENU only uses a normal HTML anchor and there is no JavaScript required, it is extremely simple to implement and probably will work with almost all browsers and devices.

The jump to the footer can be disorienting because the screen abruptly changes from one state to another without much feedback of what happened. You can decrease the problem caused by the sudden jump by using an animation that scrolls through the entire page until the menu. However, depending on the length of the page and how the animation is done, you can be unnecessarily delaying the access of the user to the menu. Moreover, this type of animation can be sluggish on slower devices.

***

Figure 6.14 *Unicef Sweden* <www.unicef.se>, April 2012.

Figure 6.15 *Bagcheck* <www.bagcheck.com>, April 2012.

Related Patterns

- LINEARIZED MENU
- TOGGLE MENU

6.5 Toggle Menu

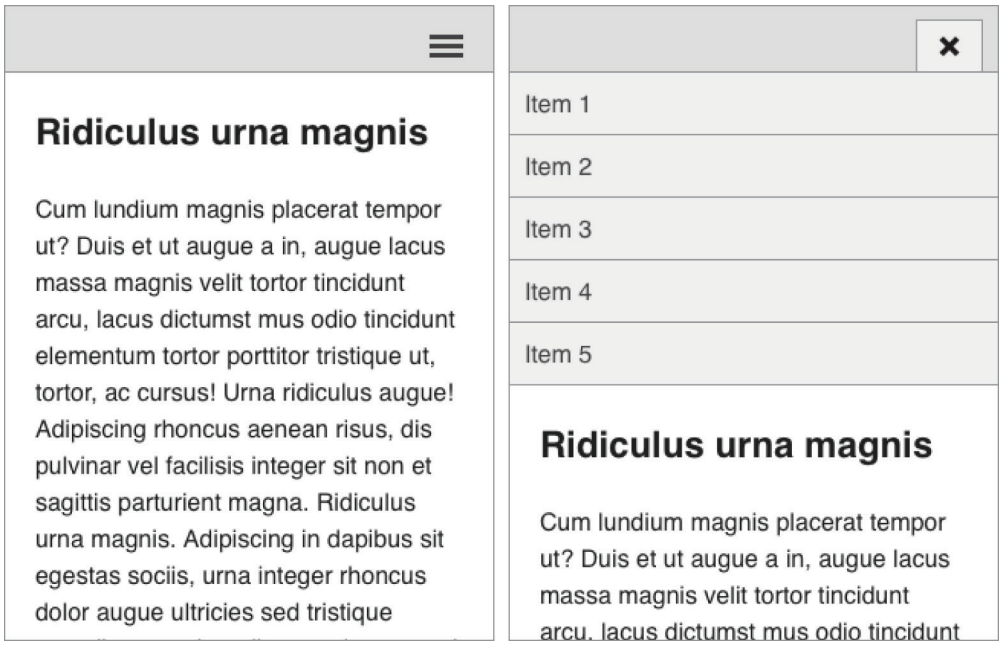


Figure 6.16 TOGGLE MENU <patterns.jribeiro.org/patterns/toggle-menu>.

Problem

You have an extensive LINEARIZED MENU that takes the entire screen, and a JUMP MENU is not a proper alternative because it displaces the menu to the bottom of the page. You want to display the menu on the top of the page but do not want it to fill the entire screen.

Solution

Design the menu content as a LINEARIZED MENU but conceal it, then provide a button to toggle the visibility of the menu.

***

In the TOGGLE MENU we have a LINEARIZED MENU that is presented collapsed when the page loads until there is a direct action of the user to expand it. This allows us to display only a small button on top of the page to toggle the visibility of the menu. With this approach we can present the content first while still providing quick access to the navigation.

Figure 6.17 A possible toggle icon.

For the element which triggers the menu you can use any symbol that provides the correct affordance that additional information will be disclosed. Nonetheless, many websites use an icon with three horizontal bars, which represent the list items of a menu (Figure 6.17). When the menu is active you can change the icon to show that the menu state has changed. For example, the Starbucks website changes the icon to an "x" when the menu is expanded. Andy Clarke (2012) incites the need to reach a consensus on a standard icon for showing navigation, settling his support for the three lines because they are widely used and therefore, easily recognized, unless your navigation is arranged on a grid, which, in that case you should use a grid icon. In short, the symbol used should map the layout of the menu.

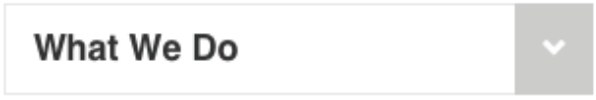


Figure 6.18 *Filament Group* <www.filamentgroup.com/examples/rwd-nav-patterns> approach to this pattern uses the toggle button to display the title of the current page.

If you have enough horizontal space you can improve the usability of this pattern by appending the title of the current page to the toggle icon (Figure 6.18). This allows us to give feedback of the user's position in the site hierarchy and provide a larger target.

***

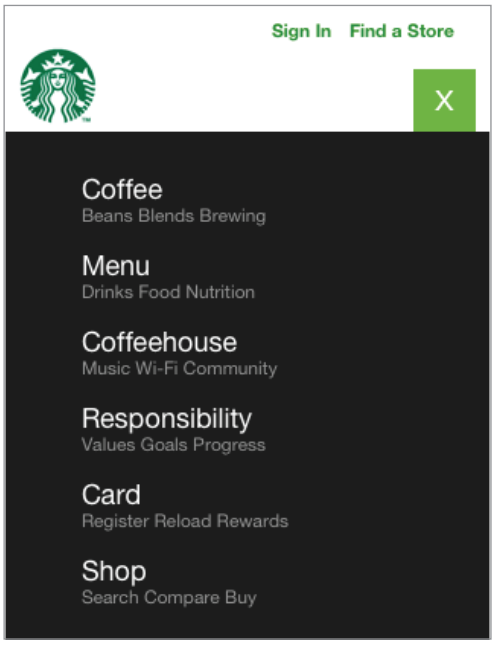


Figure 6.19 *Starbucks* <www.starbucks.com>, April 2012.

Figure 6.20 *Twitter Bootstrap* <www.twitter.github.com/bootstrap>, April 2012.

Related Patterns

- EXPANDING LIST
- SELECT MENU
- DROPDOWN

6.6 Side Menu

Figure 6.21 SIDE MENU <patterns.jribeiro.org/patterns/side-menu>.

Problem

Although vertical menus side by side with the main content are fairly common, they are almost impossible to achieve on a mobile device. It would not be efficient to reserve an entire column on a small screen just for the menu. Nonetheless, you may still want to get a similar look.

Solution

Design the menu bonded together to one side of the page, but positioned outside the page. Then, provide a button that will show the menu by sliding it in, sliding out the content.

***

Vertical navigation placed side by side with the page's main content is a quite common pattern on websites. On mobile, that type of navigation is not practical because horizontal space is limited, but this pattern allows us to achieve a comparable layout on both versions. Like the TOGGLE MENU, this pattern allows us to focus on the content while providing quick access to the navigation.

The idea of this pattern was first formulated as a pattern by Frost (2012) as *The Left Nav Flyout*, and consists of a button on top of the page that allows users to toggle the visibility of a hidden menu. When that button is tapped it reveals the menu on one of the side of the screen by pushing the main content out of the screen. You should keep a small portion of the page to

give some affordance of how the menu works. Additionally, you can display an animation of the menu moving to show users what is happening. Like in the JUMP MENU, an abrupt change of the context can disorient the user.

***

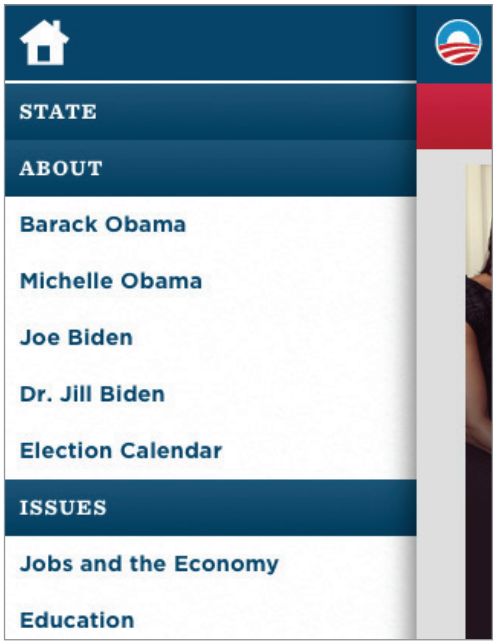


Figure 6.22 *Barack Obama* <www.barackobama.com>, April 2012.

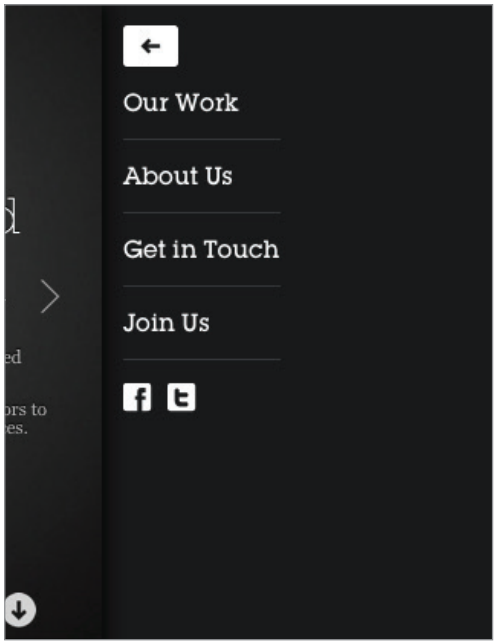


Figure 6.23 *Kettle* <www.kettlenyc.com>, April 2012.

Related Patterns

- TOGGLE MENU

6.7 Select Menu

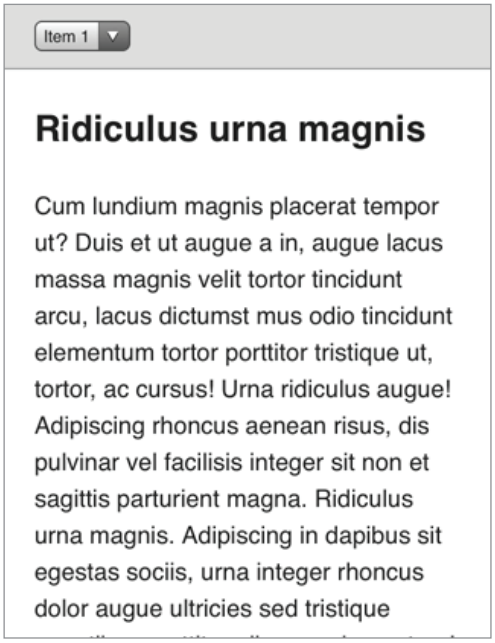


Figure 6.24 SELECT MENU <patterns.jribeiro.org/patterns/select-menu>.

Problem

On small screens, an extensive menu can fill the entire page. However, if you have a menu that has to work simultaneously on wide screens and on mobile devices, you want that menu to be as efficient as possible regarding the use of vertical space.

Solution

Present the navigation on a menu that on narrow screen devices dynamically changes to a native select component.

***

This pattern is useful for designing a navigation menu with numerous and lengthy items that needs to work simultaneously on the mobile and desktop version. In the desktop version of the website the menu is presented expanded, on a narrow screen it is converted through JavaScript to the native select component. Thus, this pattern is normally seen in websites that implement a responsive design.

This type of menu can be a practical alternative for when vertical space is scarce and you want to display the menu on top of the page. However, it is not the most elegant of the alternatives, because it adds another layer of information with a distinctive interface. A more clean

and elegant approach in terms of visual design can be achieved through the TOGGLE MENU pattern, which uses a comparable type of interaction but with a custom interface.

This type of menu is easier to recognize as something selectable because it uses the native controls of each device. Likewise, because it uses controls that are optimized for the respective device, you can be confident that it will work and be accessible in most of them. Though, as a downside, because it uses the native browser components, it is very difficult to achieve a consistent look across platforms (Figure 6.25).

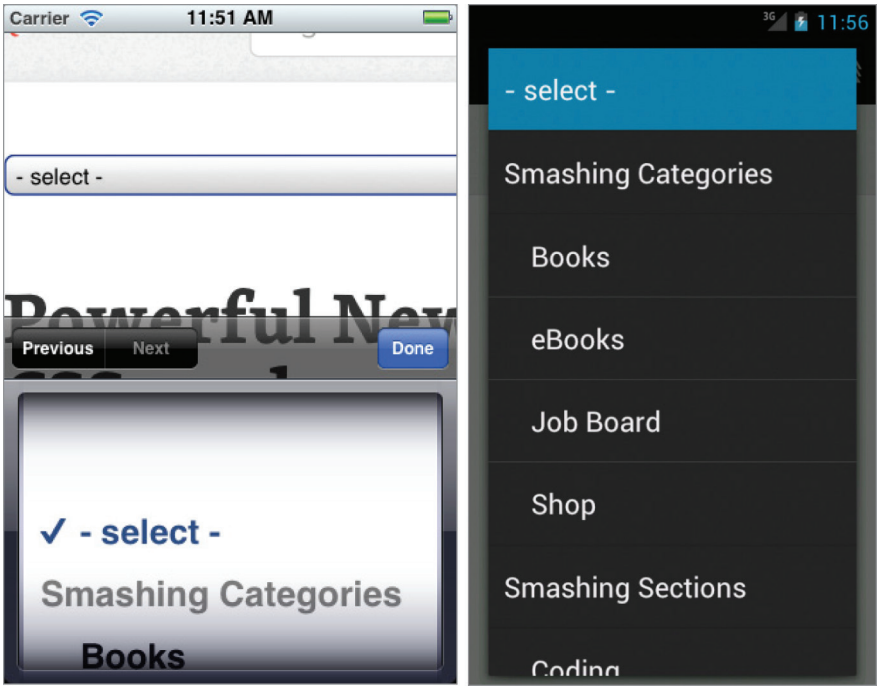


Figure 6.25 Screenshots from the same SELECT MENU at *Smashing Magazine* website <www.smashingmagazine.com> on different platforms; iOS on the left, Android on the right.

You can also work with subitems, though that can be even more confusing. In Figure 6.25 subitems are denoted by an indent, but dashes are also a common alternative.

***

Figure 6.26 *WorldSkills London 2011* <www.worldskillslondon2011.com>, April 2012.

Figure 6.27 *Lancaster University* <www.lancs.ac.uk>, April 2012.

Related Patterns

- TOGGLE MENU

6.8 Vertical List

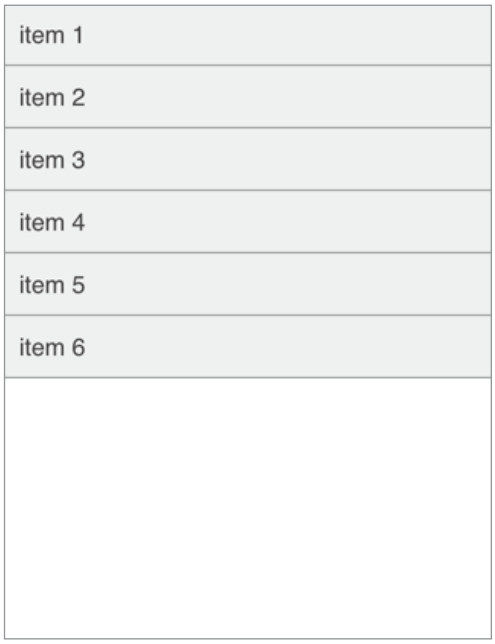


Figure 6.28 VERTICAL LIST <patterns.jribeiro.org/patterns/vertical-list>.

Problem

A LINEARIZED LAYOUT is a quite common layout that gives much emphasis to the vertical orientation. If you have information that is organized as a list, you need to have a method to efficiently display that content of this type of layout.

Solution

Display these chunks of information stacked vertically, spanning all the width of the screen and graphically dividing each item.

***

Along with the LINEARIZED LAYOUT this is perhaps the most common pattern that you will encounter when designing for mobile. Given that the width of the devices is not generous, you will invariably need to rely on the vertical space to accommodate most of the content. For most of these situations, this pattern or one of its variations will be used, such as: INFINITE LIST, THUMBNAIL LIST or EXPANDING LIST. Therefore, most of the recommendations given for this pattern also apply to the related patterns.

Generally, there are two different alternatives to the implementation of this pattern: one that only displays information; another where each list item works as a link — which sometimes can be a LINEARIZED MENU.

When the list items work as links you should optimize them for touch — you can use a TOUCH FRIENDLY TARGET for this. Each item ideally should only contain a link. Even if the item contains text with different hierarchies (Figure 6.29), you can wrap all those elements on an anchor tag rather than using only the title as a link. The result is similar to what can be attained with an ICEBERG TIP.

Figure 6.29 On the *Authentic Jobs* website <www.authenticjobs.com>, while each list item contains a diverse range of information with different hierarchies, the entire rectangle works as a link.

For this pattern to work effectively, each item should be clearly separated. You can use any design that visibly distinguishes items, such as: a horizontal line spanning the width of the screen; alternative color rows; typographic hierarchy; or just white space.

***

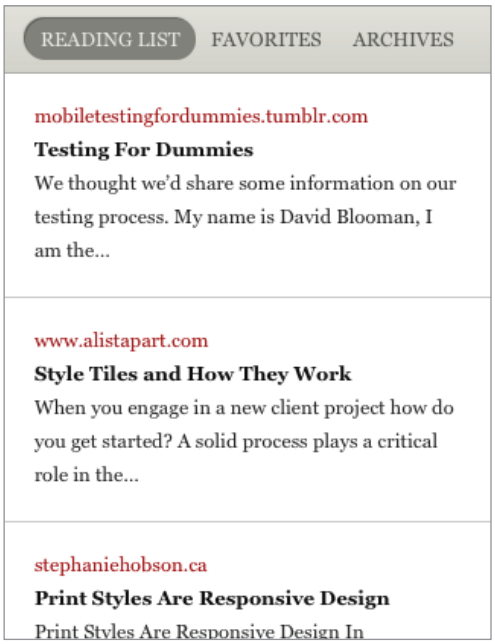


Figure 6.30 *Readability* <www.readability.com/mobile>, April 2012.

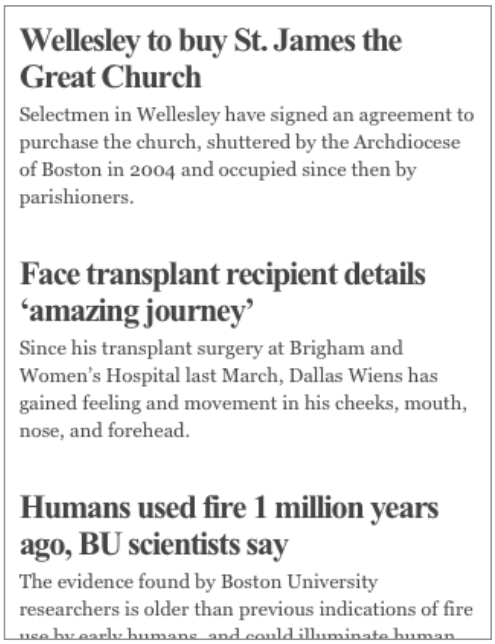


Figure 6.31 *Boston Globe* <www.bostonglobe.com>, April 2012.

Related Patterns

- INFINITE LIST
- THUMBNAIL LIST
- EXPANDING LIST
- LINEARIZED MENU

- *Vertical List* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)
- *List Menu* in *Mobile Design Pattern Gallery* (Neil 2012)

6.9 Infinite List

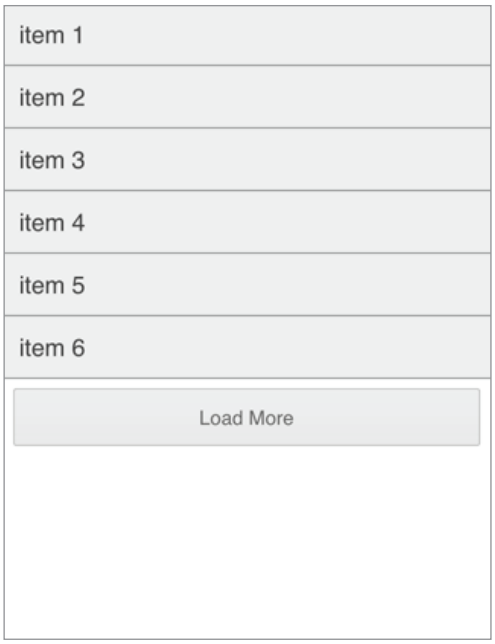


Figure 6.32 INFINITE LIST <patterns.jribeiro.org/patterns/infinite-list>.

Problem

A very extensive VERTICAL LIST can become quite heavy on mobile devices. It would not be advisable to load and display all information in the list at once.

Solution

Design a normal VERTICAL LIST or any of its variations but only fetch the beginning of the list, loading the rest as the user scrolls through the page.

***

This pattern is based on a homonym pattern from *Designing Mobile interfaces* (Hoober and Berkman 2011). It is very similar to the VERTICAL LIST with the main difference that only a portion of the list is initially loaded. It is most adapted for displaying a list of data with an uncountable number of items. For example, when loading all the news of a site in the same page we can be dealing with hundreds of items. It would not be a good idea to load all that information at once. That would be extremely heavy in terms of download and load time; it would add an unnecessary burden in terms of browser memory and rendering time; and probably, users would not need all that information from the beginning. Thus, we can start by loading only a few items, and only load subsequent items when the user provides some signal that he wants

to keep browsing through the page. The number of items to load on each action varies, but it will depend on the length and download size of the content.

This pattern takes advantage of methods for asynchronously loading additional content without the need to refresh the page: new items are just appended to the interface following the previous ones. Because the browser does not need to reload the page, we can get a perceivably faster response. It can be used as an alternative to a common pagination, which normally implies more touches and a refresh on each load. Since there is not any refresh, users never lose the context of where they are at the list.

There are two main alternatives that can be used for its implementation: explicit and implicit loading. On the first, the content is loaded with a direct action of the user; on the second, the list is loaded automatically as the user reaches its end.

When designing a list in which the user has to explicitly load new content, place at the bottom of the list a button that indicates that more data will be loaded on the same page. You can use a label with "more", "load more", or another appropriate variation. You can also use that button to indicate how many more items will be loaded.

While the browser is loading the next chunk of the list, provide some feedback of the action that is occurring. In Figure 6.33, the page displays a 'More' button that when pressed changes to a loading animation.

Figure 6.33 Loading animation at *The Verge* <www.mobile.theverge.com>.

In the implicit loading mode there is not any direct action of the user to load additional content. Instead, the browser detects when a user reaches the end of the page and automatically loads another portion of the list. This is a type of implementation that is normally called *lazy loading*. To give the impression that the list is really bottomless, you can start preloading the adjacent content before the user gets to end of the list.

Although this pattern is normally used with content that is virtually endless, it is still possible for users to reach its end. Therefore, provide some indication when there are no more items to fetch.

Because this pattern is a variation of the VERTICAL LIST, it can be combined with any of its other variants, such as: THUMBNAIL LIST, EXPANDING LIST.

***

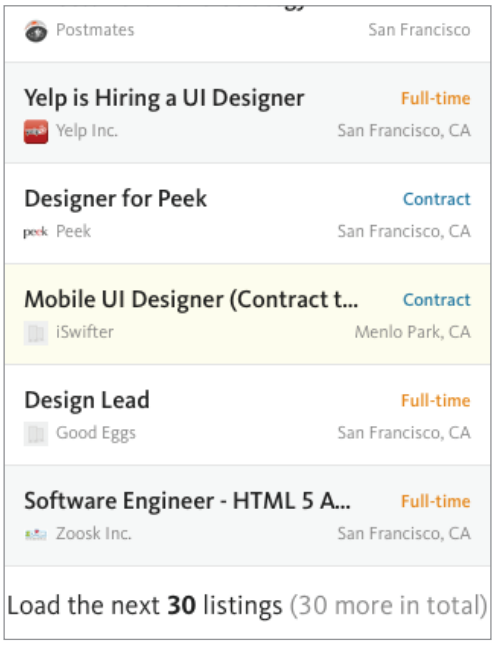


Figure 6.34 *Authentic Jobs* <www.authenticjobs.com>, April 2012.

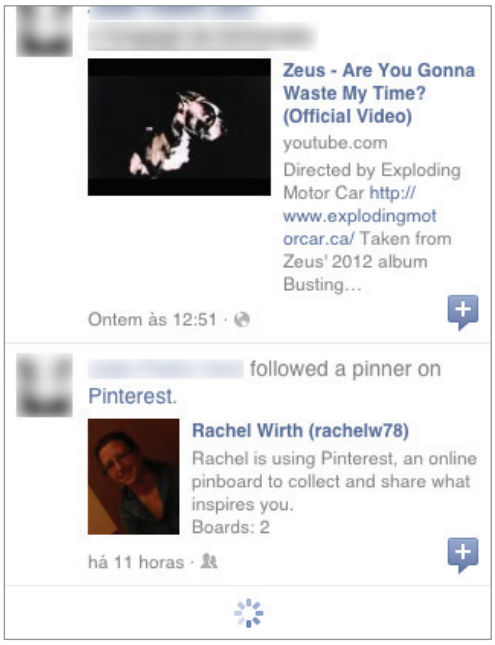


Figure 6.35 *Facebook* <www.facebook.com>, April 2012.

Related Patterns

- VERTICAL LIST
- THUMBNAIL LIST
- EXPANDING LIST

- *Infinite List* in *Designing Interfaces* (Tidwell 2011)
- *Infinite List* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)

6.10 Thumbnail List

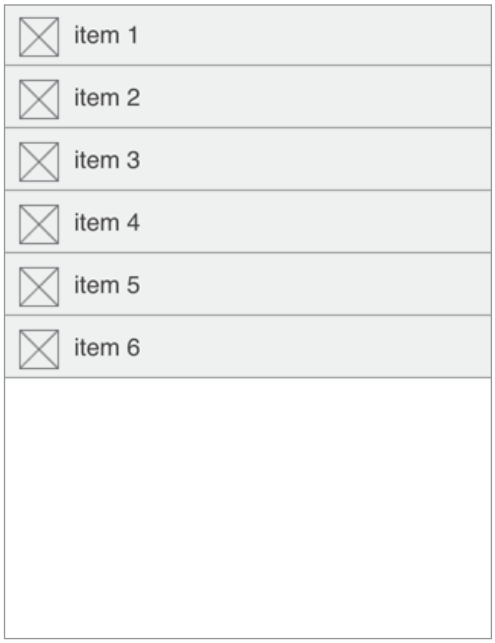


Figure 6.36 THUMBNAIL LIST <patterns.jribeiro.org/patterns/thumbnail-list>.

Problem

It can be difficult to find a particular item on a VERTICAL LIST that is composed only by text because all items may look identical. You need to make each list item more distinct so that the list is easier to scan.

Solution

Design a VERTICAL LIST where in addition to the textual information, you also display a small illustration next to each list item.

***

An extensive list of items composed only of text can be visually monotonous and harder to scan. You can minimize this problem by complementing each list item with a thumbnail-size image that is illustrative of the content. Because each image can have distinct shapes and colors, they are easier to scan and interpret. The image should somehow be related to the content of the item, but you can use either a photo or an icon. This pattern is based on the patterns *Thumbnail List* (Hoober and Berkman 2011), from where it got its name, or *Thumbnail-and-Text List* (Tidwell 2011).

Thumbnails are usually aligned to the left. However, if images are optional, you can align them to the right so you can create a better defined axis (Figure 6.28). You can use a placeholder image for instances where images are not available, but keep in mind that if most images are placeholders the benefits of the THUMBNAIL LIST are lost.

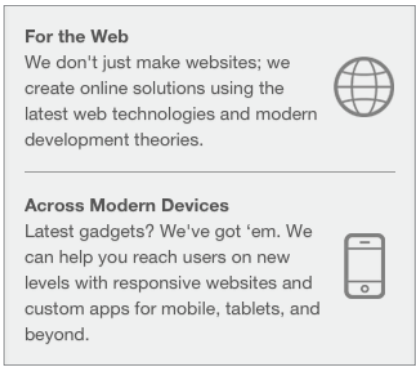


Figure 6.37 Right aligned THUMBNAIL LIST at *Meltmedia* website <www.meltmedia.com>.

This pattern can be used with the other variations of the VERTICAL LIST, such as the INFINITE LIST and the EXPANDING LIST.

***

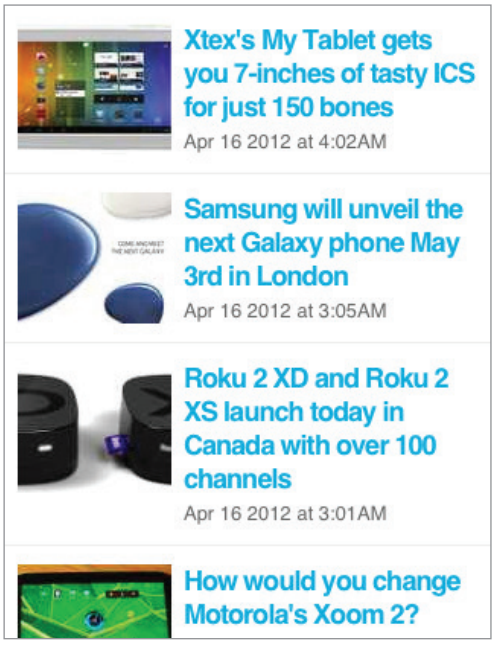


Figure 6.38 *The Verge* <www.mobile.theverge.com>, April 2012.

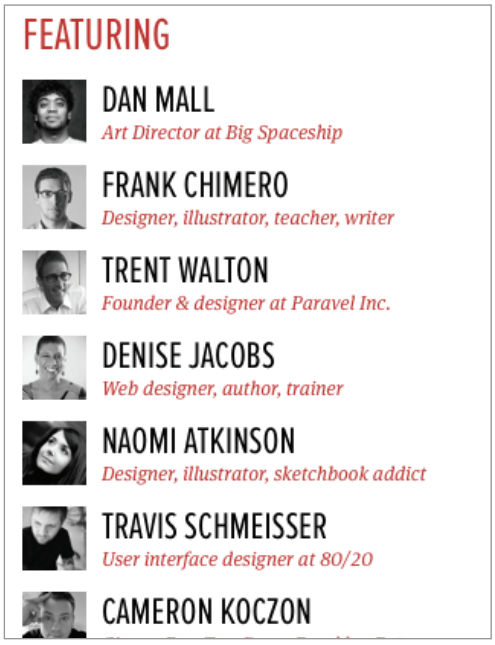


Figure 6.39 *New Adventures* <www.2012.newadventuresconf.com>, April 2012.

Related patterns

- VERTICAL LIST
- INFINITE LIST
- EXPANDING LIST
- LINEARIZED MENU

- *Thumbnail List* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)
- *Thumbnail-and-Text List* in *Designing Interfaces* (Tidwell 2011)

6.11 Expanding List

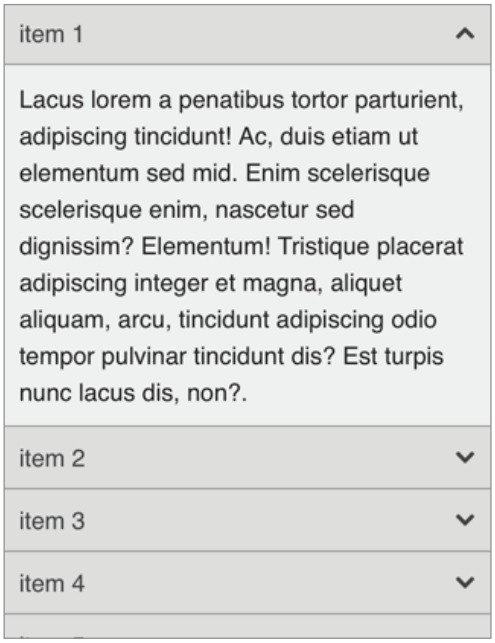


Figure 6.40 EXPANDING LIST <patterns.jribeiro.org/patterns/expanding-list>.

Problem

You need to display a series of related information that has a clearly defined hierarchy. However, vertically displaying all that information would lead to a very long page.

Solution

Design a VERTICAL LIST or one of its variations, but display only part of the content — usually the heading — as a toggle to show additional content.

***

This pattern got its name from a similar pattern in *Mobile Design Pattern Gallery* (Neil 2012) and is a variation of the VERTICAL LIST, in which the visible item does not present static information or works as a link to another page, but is rather used to trigger the visibility of additional content in the same page. Tapping on the visible part of the item makes it expand, revealing the hidden content. It is most suitable for when you need to present content with a clearly defined hierarchy; for example, when you are designing a LINEARIZED MENU with subitems.

Although it is possible to present more than two levels of information with this pattern, it can be confusing to do so.

You should provide some clues to indicate that additional content is available. For example, a downward arrow that changes to an upward arrow when the item is expanded; or a plus sign

that changes to a minus sign. You can give emphasis to the fact that the item has expanded by implementing a small animation showing the content appearing.

Besides clearly distinguishing between list items — as it is described on the VERTICAL LIST —, you should also differentiate between the heading of the item and its respective content. More important, you should design them so that the revealed content is grouped to the upper heading rather than the order way around.

Interactions Details

In terms of the behavior of the list there are two alternatives for the implementation of this pattern: one that works as a toggle; another that works as an accordion. In the toggle type each item works independently, that is, regardless of the state of all other items in the list, when you tap on one it expands, when you tap it again it collapses. In the accordion type, elements of the list are connected, when the user taps on the header of the item the content of that item is expanded and all others are collapsed. These two different behaviors are sometimes described as different patterns, for example, in *Designing Interfaces* (2011), Tidwell presents the patterns, *Accordion*, and *Collapsible Panels*.

***

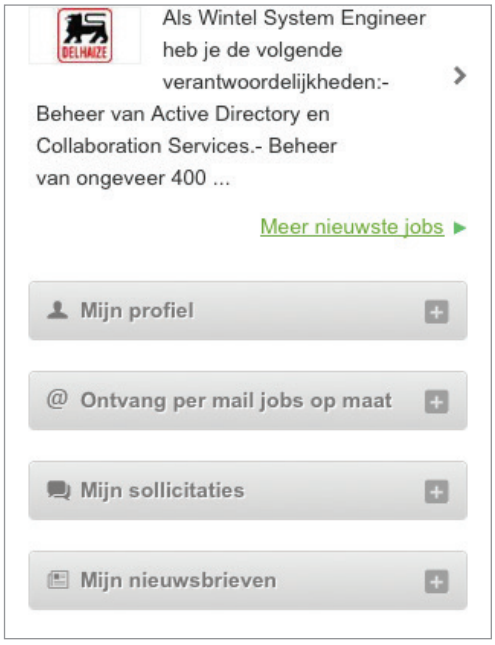


Figure 6.41 *Jobs At* <www.jobat.be>, April 2012.

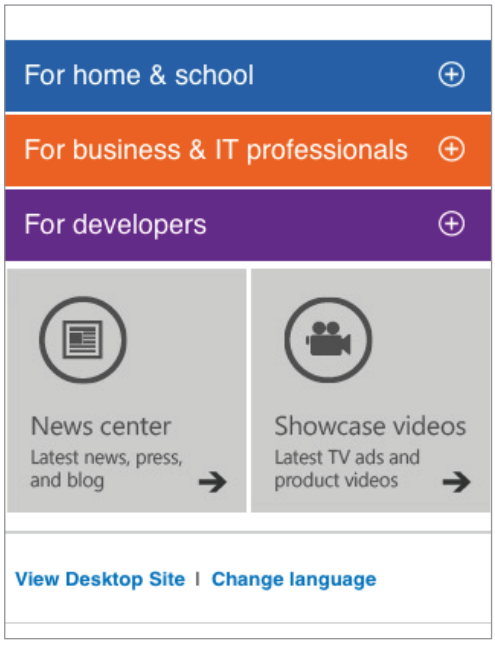


Figure 6.42 *Microsoft* <www.m.microsoft.com>, April 2012.

Related Patterns

- VERTICAL LIST
- INFINITE LIST
- THUMBNAIL LIST
- LINEARIZED MENU

- *Expanding List* in *Mobile Design Pattern Gallery* (Neil 2012)
- *Windowshade* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)
- *Accordion* in *Designing Interfaces* (Tidwell 2011)
- *Collapsible Panels* in *Designing Interfaces* (Tidwell 2011)

6.12 Fixed Content

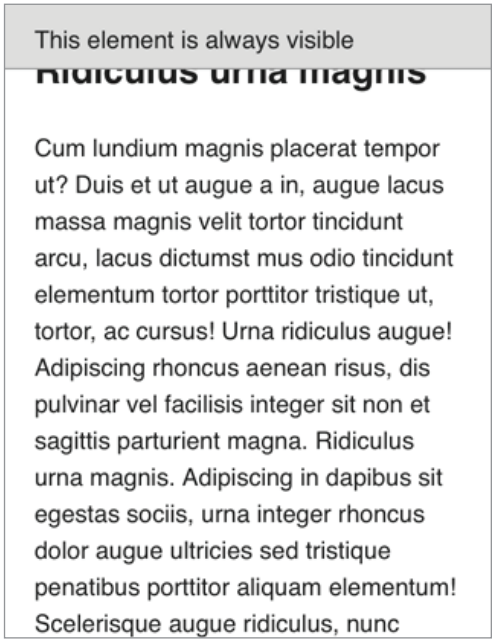


Figure 6.43 FIXED CONTENT <patterns.jribeiro.org/patterns/fixed-content>.

Problem

The normal behavior of content on a page is to go off the viewport as the user scrolls through. However, you may need to provide quick access to functions or information persistently through the entire page.

Solution

Present the content positioned fixed to the edges of the browser window, over the page, and assuring that it is visible through the entire page.

***

FIXED CONTENT allows us to provide quick access to functions that need to be present through the entire page, or alert the user of some important information. Given that, it is commonly used for designing web applications or to give a more native look to the interface.

A major drawback[40] with this pattern is that it takes a considerable amount of space, which is already a limited asset on these devices. Besides of the already small height of the device,

40 Another downside is caused by a somehow buggy implementation of the position fixed on current mobile browsers. In some older browsers it simply just does not work, but it is especially problematic on the browsers that implement it poorly rather than those where it is just ignored. Thus, do not assume that your design will work, without carefully testing it. You can

we have to account for the OS toolbar, the browser chrome, and a potential keyboard, all those contributing for reducing the effective real estate of the page. This problem is even more prevalent when the device is oriented in landscape. Therefore, make sure that any FIXED CONTENT is absolutely essential in your website.

***

Figure 6.44 *dConstruct 2011* <www.2011.dconstruct.com>, April 2012.

Figure 6.45 *Twitter* <www.mobile.twitter.com>, April 2012.

Related Patterns

- *Bottom Navigation* in *Designing Interfaces* (Tidwell 2011)
- *Fixed Menu* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)

---

solve some of these problems through JavaScript, though, by doing that you are adding complexity and it will probably still not work with some older browsers.

6.13 Slideshow

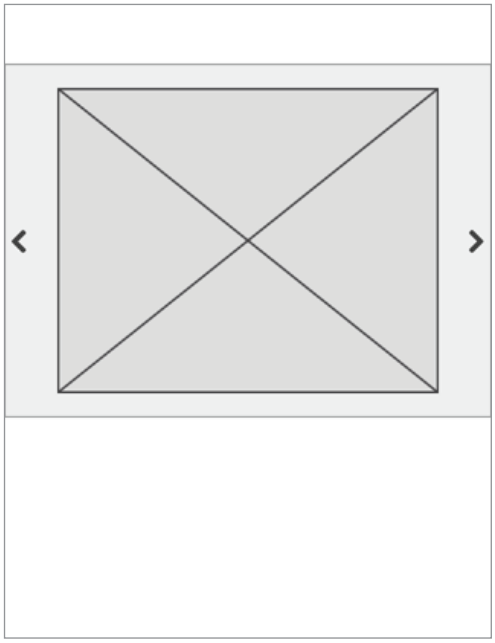


Figure 6.46 SLIDESHOW <patterns.jribeiro.org/patterns/slideshow>.

Problem

You need to display a series of related information, with comparable content in terms of length, without consuming a considerable amount of vertical space.

Solution

Reveal items one at a time, by changing the visible item automatically in a specific time interval or by providing a method for browsing through the entire content.

***

When you have a series of related content that is extensive but not considerably important, you can save vertical space by having all pieces of that content arranged horizontally, placed virtually beyond the width of the device, while keeping a window — generally with the same width of the device — that works as a viewfinder for the list. Users can only see one item of the list at each time but have some method for traversing through the list. The SLIDESHOW is a common pattern on the web, and was previously formulated as a pattern for mobile in *Designing Mobile Interfaces* (Hoober and Berkman 2011). It is commonly used with images, although it can be successfully implemented with text, or image and text. Each slide can be simply used to display information but can also work as a link.

However, do not use this pattern for presenting critical information. Since only one item will be visible at any time and users may not understand that they can scroll through the list, hidden content can easily go unnoticed. The SLIDESHOW is most suitable for presenting more casual information like a gallery of images.

Although you can use slides with different content in terms of length, it is a good idea to keep the slide height constant across slides to prevent the layout from moving up and down between slides.

Interaction Details

Signalize that additional content is hidden, moreover, use that sign as a hint of how to reveal it. An arrow or an index of the list (Figure 6.47) are commonly used to solve this problem. Alternatively, you can display a portion of the previous and following images to alert users that more content is available — closer to the behavior of what is usually described as a *carousel*.

For scrolling through the content you can use one of these approaches or a combination of them:

- Slides change automatically without any control of the user.
- A tap on the slide to make it move to the next one; if this is the only method for navigating the SLIDESHOW, it has the inconvenient that if you have a long list and want to go back one slide, you need to scroll through the entire list.
- You can use a "next" and "previous" buttons for scrolling through the slideshow; arrows or textual descriptions are usually used for this.
- Use a swipe gesture, which is probably the most elegant and a natural of the alternatives because you are directly manipulating the content; however, because of its relative novelty it is harder to be discovered and more difficult to implement.Whenever possible try to take advantage of the swipe gesture to traverse trough the gallery, but it is a good practice to provide a fallback — a button — for devices that may not support gestures; and for users who may not understand it or are not used to this kind of interaction.

Nevertheless, implement an animation between each slide to help users grasp what is happening. A crossfade is common with this type of pattern, but an animation of the section sliding in and out can provide a better affordance, particularly if a swipe is used to move between slides.

Figure 6.47 SLIDESHOW index.

Provide some feedback of the user's position within the list. It can be done by using an index of the list (Figure 6.47), and if it is important to identify each slide, you can number them. Markers can also work as buttons that link to the corresponding slide, although they need to be large enough to work efficiently. Thus, you should always provide an alternative method for scrolling through the list.

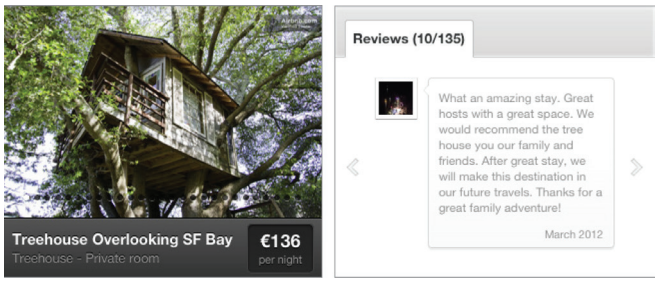


Figure 6.48 Screenshots from *Airbnb's* product page <www.m.airbnb.com>.

The *Airbnb* website has two SLIDESHOW galleries in the same page, and each one implements a different method for browsing the gallery: one only works with a swipe (Figure 6.48, left image) and one only works with buttons (Figure 6.48, right image). Although they are used for different purposes and are visually distinct, it can still confuse users and it would be expectable that at least the interaction worked identically.

***

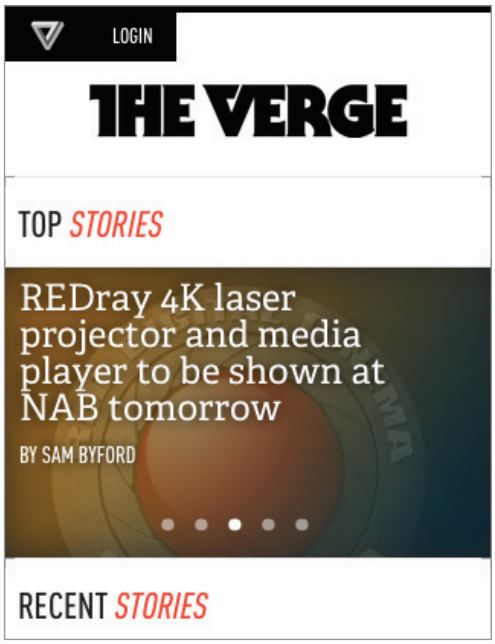


Figure 6.49 *The Verge* <www.mobile.theverge.com>, April 2012.

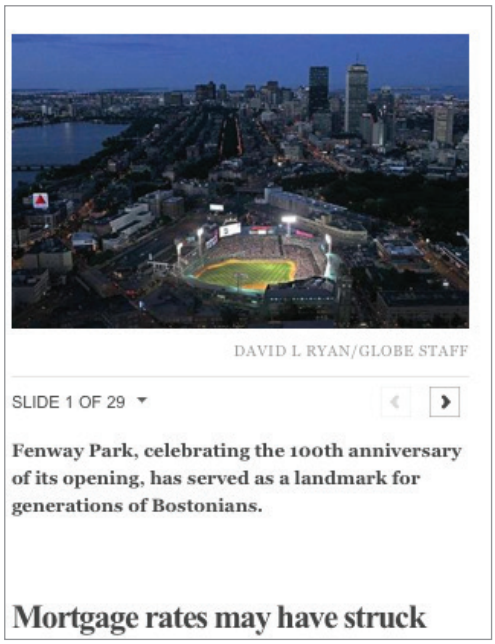


Figure 6.50 *Boston Globe* <www.bostonglobe.com>, April 2012.

Figure 6.51 *Zappos* <www.m.zappos.com>, April 2012.

Related Patterns

- *Slideshow* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)

6.14 Tabs

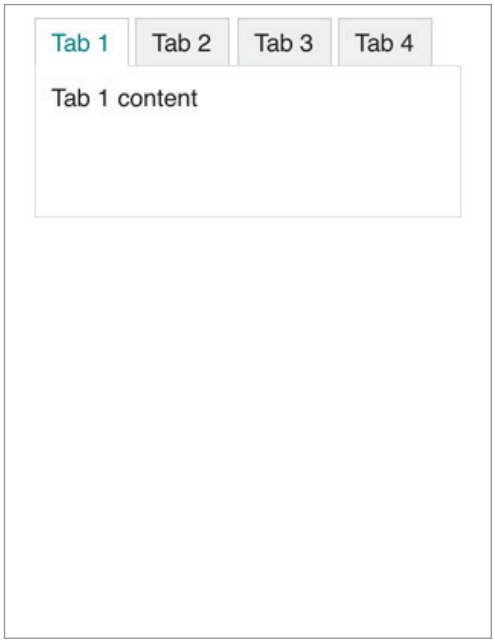


Figure 6.52 TABS <patterns.jribeiro.org/patterns/tabs>.

Problem

You have a series of related information with a clearly defined hierarchy, and comparable length that needs to be presented at a similar level without wasting too much vertical space.

Solution

Arrange horizontally the headings of all items of those elements, but display only the content of one. The visibility of each one can be toggled by users.

***

TABS are widely used in web design and are therefore recognizable by users. They use a metaphor of the labels on folder archives which make them easy to understand. TABS should be used to alternate between views within the same context (Nielsen 2007) rather than between pages. That is, they do not take the user to another page, only the visibility of the content changes.

You should clearly identify which is the active tab. The tab heading should be visually connected to the content for better making this distinction. You should take special attention when there are only two tabs, because the inactive state can be more easily mistaken as being active.

TABS work better when you only have a few tab modules that fit on the width of the page, and there is only one row of tabs. Two rows or more of tabs are a quite confusing and not very elegant. If you have a list of tabs headings that do not fit the width of the device it is better to

truncate part of the list and to provide a method for scrolling through the tabs headings — a behavior similar to a SLIDESHOW. Nonetheless, it may be a better idea to revise your design and think of an alternative approach. A VERTICAL LIST or a SLIDESHOW can sometimes be an option.

***

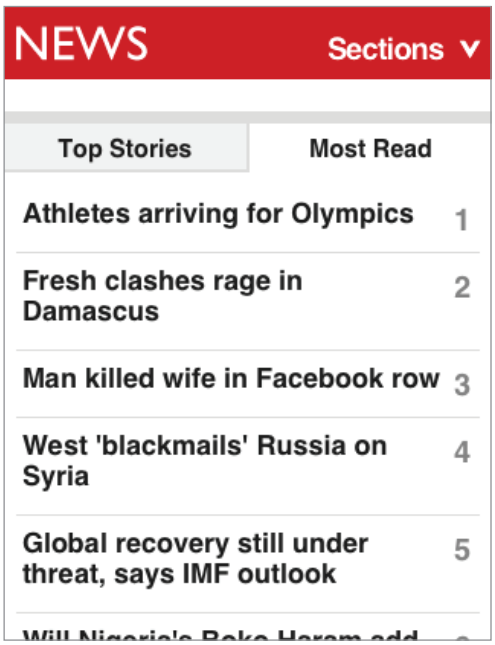


Figure 6.53 BBC <www.m.bbc.co.uk/news>, July 2012.

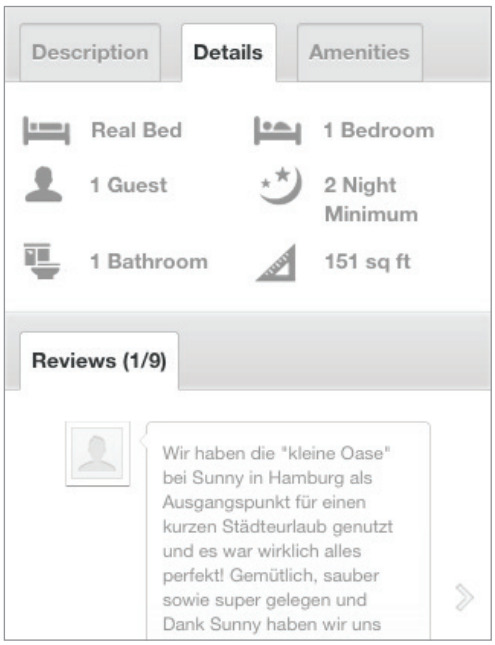


Figure 6.54 *Airbnb* <www.m.airbnb.com>, April 2012.

Related Patterns

- SLIDESHOW

- *Tabs* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)
- *Tab Menu* in *Mobile Design Pattern Gallery* (Neil 2012)

6.15 Dropdown

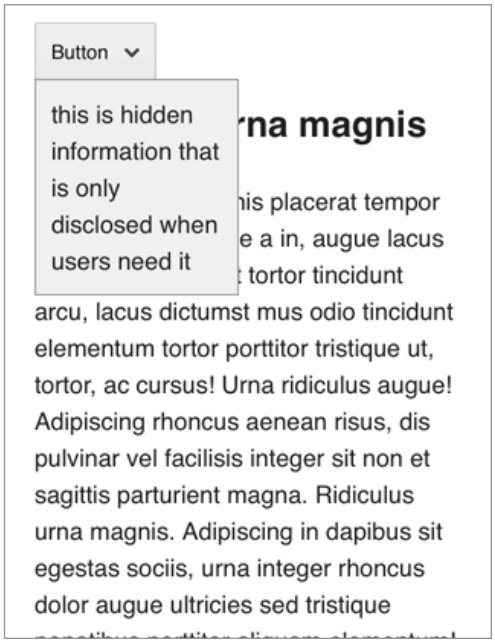


Figure 6.55 DROPDOWN <patterns.jribeiro.org/patterns/dropdown>.

Problem

Sometimes you may have information that is not frequently needed. To simplify the interface it would be convenient to remove it; however, it is still important to provide access to that content.

Solution

Design an element on the page that toggles the visibility of additional content. Keep the content hidden until the users express a direct intention to access it, then, make the content appear over the page.

***

You should try to not overload the page and the user with information that is not frequently needed. A DROPDOWN allows you to keep the layout simpler and cleaner by concealing non-essential information until there is a direct action of the user. You can use it to present small pieces of information that do not exceed the height of the screen. That is, you should not add additional complexity to this interface component by having the user scroll the page or the DROPDOWN to see truncated content.

In a DROPDOWN you have a button or any other element on the page that once tapped reveals the hidden content hovering on top of the page. Users should be able to withdraw it by tapping the same button again or any part of the page that is not the DROPDOWN.

***

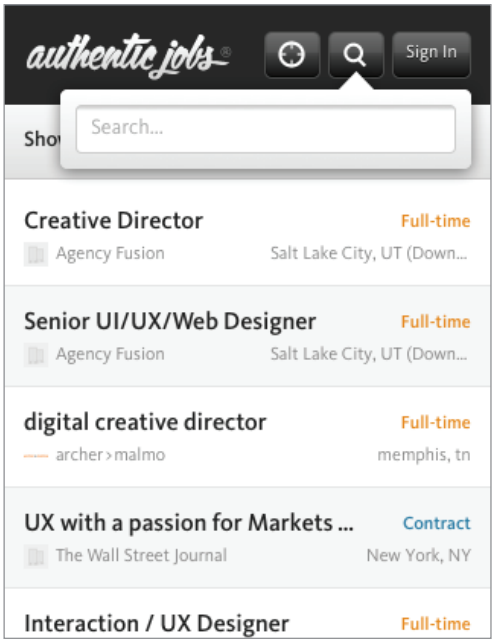


Figure 6.56 *Authentic Jobs* <www.authenticjobs.com>, April 2012.

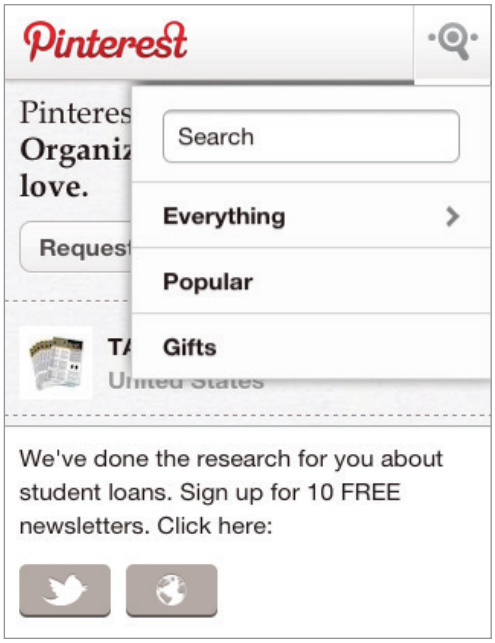
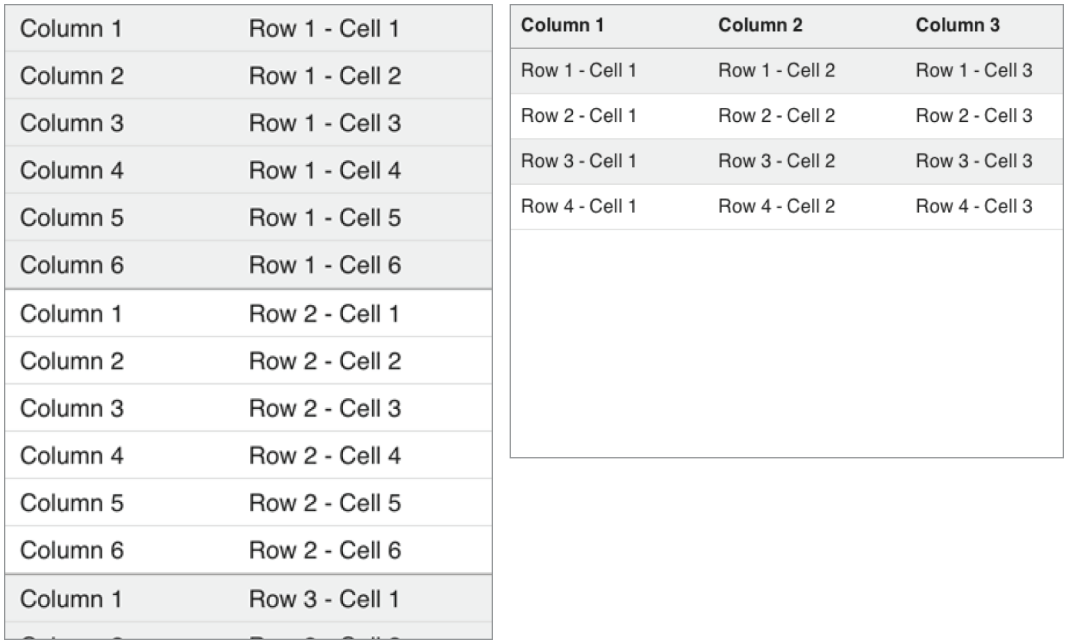


Figure 6.57 *Pinterest* <www.m.pinterest.com>, April 2012.

Related Patterns

- EXPANDING LIST

6.16 Linearized Table



Column 1	Row 1 - Cell 1
Column 2	Row 1 - Cell 2
Column 3	Row 1 - Cell 3
Column 4	Row 1 - Cell 4
Column 5	Row 1 - Cell 5
Column 6	Row 1 - Cell 6
Column 1	Row 2 - Cell 1
Column 2	Row 2 - Cell 2
Column 3	Row 2 - Cell 3
Column 4	Row 2 - Cell 4
Column 5	Row 2 - Cell 5
Column 6	Row 2 - Cell 6
Column 1	Row 3 - Cell 1

Column 1	Column 2	Column 3
Row 1 - Cell 1	Row 1 - Cell 2	Row 1 - Cell 3
Row 2 - Cell 1	Row 2 - Cell 2	Row 2 - Cell 3
Row 3 - Cell 1	Row 3 - Cell 2	Row 3 - Cell 3
Row 4 - Cell 1	Row 4 - Cell 2	Row 4 - Cell 3

Figure 6.58 LINEARIZED TABLE <patterns.jribeiro.org/patterns/linearized-table>.

Problem

Wide tables do not fit seamlessly on small screens. If you try to design a table with a considerable number of columns you will end up with a horizontal scroll on the page.

Solution

Linearize the table by converting each table row to its own table with two columns: one for the headings, another for the cells.

***

Tables can be quite wide, which is a problem on small screens. You can scale them down until they fit the screen, but that makes the text unreadable; or you can display them at normal size, but that leads to horizontal scrolling. Both alternatives are far from being desired. To overcome this problem you can reformat tables to a more linear design, in which table rows become independent entities stacked on top of each other. In this new adapted design, table headings are removed and each table row is converted to its own simplified table with only two columns: one for the table headers and another for the corresponding cells. Like in a normal table, you should also use alternated colors (or any appropriate design) in each new section so they are clearly distinguished. The idea for this pattern was proposed by Chris Coyier (2011) on the article *Responsive Data Tables*.

This approach works particularly well when you have a simple table with bi-dimensional data. With more complex tables — those that have headers with two or more levels — it can be harder to clearly linearize all the information without compromising its clarity.

***

GPA	N/A
Arbitrary Data	Edlund, Ben (July 1996).
First Name	Jokey
Last Name	Smurf
Job Title	Giving Exploding Presents
Favorite Color	Smurflow
Wars of Trek?	Smurf
Porn Name	Smurflane Smurfmutt
Date of Birth	Smurfuary Smurfteenth, 1945

Figure 6.59 CSS *Tricks* <www.css-tricks.com/examples/ResponsiveTables/responsive.php>, July 2012.

Related Patterns

- LINEARIZED LAYOUT
- ABRIDGED TABLE

6.17 Abridged Table

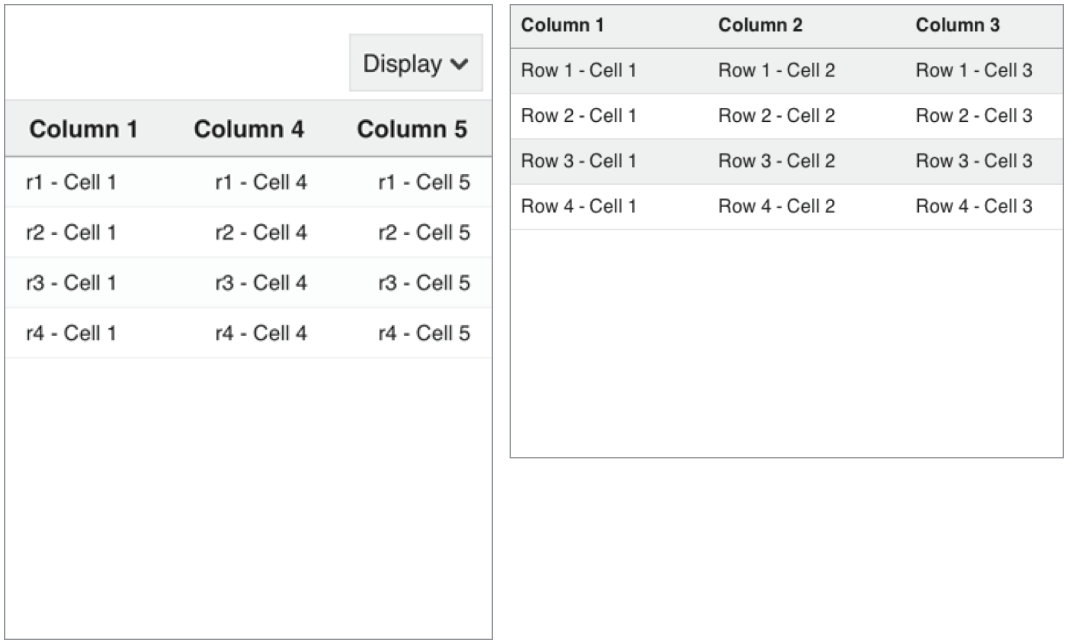


Figure 6.60 ABRIDGED TABLE <patterns.jribeiro.org/patterns/abridged-table>.

Problem

You need to present a table with several columns that does not fit on the width of a device. You could reformat the table; however, the spatial relations established on the table need to be preserved.

Solution

Display the table with some of the columns hidden, but provide a method for users to toggle the visibility of the hidden columns.

***

An ABRIDGED TABLE is an alternative to the LINEARIZED TABLE, particularly useful when the order and relations established on the table are important for its understanding. This pattern is most suitable for responsive designs because it allows us to automatically conceal columns on a table depending on the width of the device. It should be implemented in a way that permits to specify the order in which columns should be hidden, so non-essential columns can be removed first. You should also provide a method that allows users to reveal the hidden columns; a button that triggers a DROPDOWN with a list of all available columns can be a solution to this problem.

***

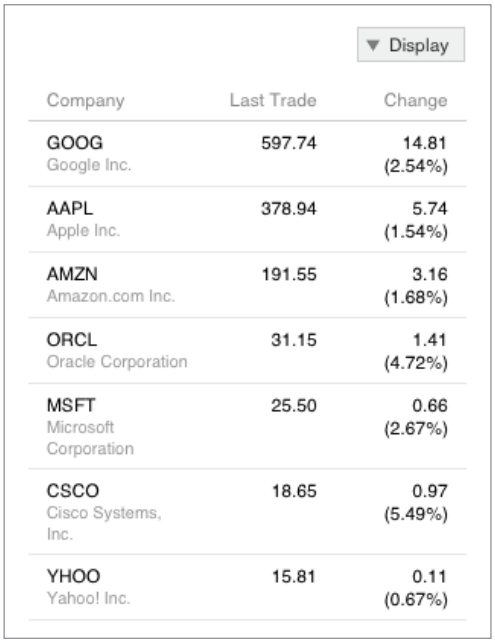


Figure 6.61 *Filament Group* <www.filamentgroup.com/examples/rwd-table-patterns>, July 2012.

Related Patterns

- LINEARIZED TABLE

6.18 Dynamic Filtering

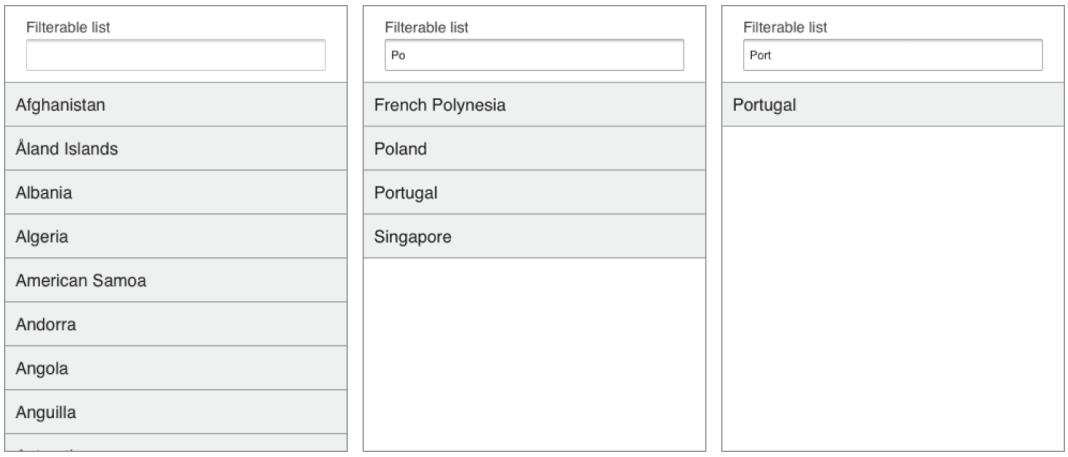


Figure 6.62 DYNAMIC FILTERING <patterns.jribeiro.org/patterns/dynamic-filtering>.

Problem

Searching through an extensive list of items within a page can be a tiresome task. Likewise, searching on a long dataset, especially if the user does not remember exactly the search term, can also be very time consuming.

Solution

Provide a search form that dynamically filters the results as the user is typing.

***

The idea for the DYNAMIC FILTERING can be found in patterns such as, *Dynamic Search* (Neil 2012) or *Search Within* (Hoober and Berkman 2011), and you can implement it on forms in order to present faster results, by minimizing the users' need to type and scroll. Unlike an explicit search that forces users to type the complete search term and press a button confirm it, with a DYNAMIC FILTERING they can get the intended result by typing only a few letters. Because users do not need to type everything, this pattern can also be helpful for cases when they do not remember accurately the desired query.

When the user types a letter the DYNAMIC FILTERING removes entries that do not contain that letter. As the user keeps typing, the system keeps eliminating entries that do not fit the pattern entered.

***

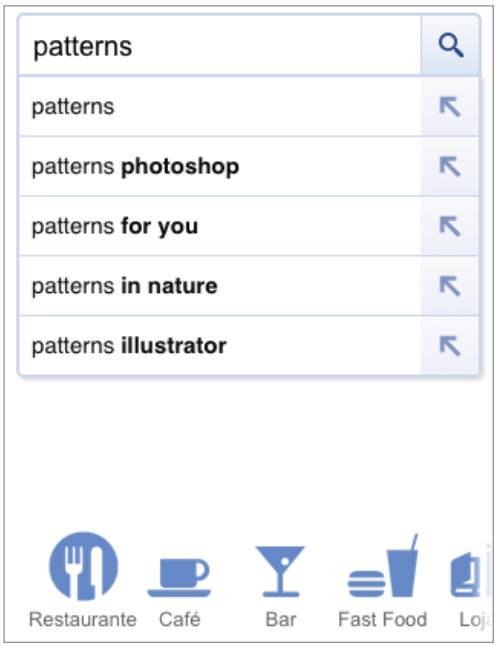


Figure 6.63 *Google* <www.google.com>, April 2012.

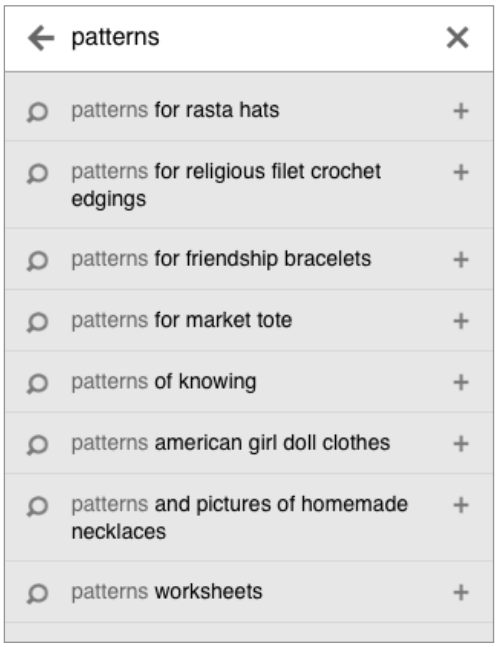


Figure 6.64 *Bing* <www.bing.com>, April 2012.

Related Patterns

- *Dynamic Search* in *Mobile Design Pattern Gallery* (Neil 2012)
- *Search Within* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)

6.19 Clear Entry

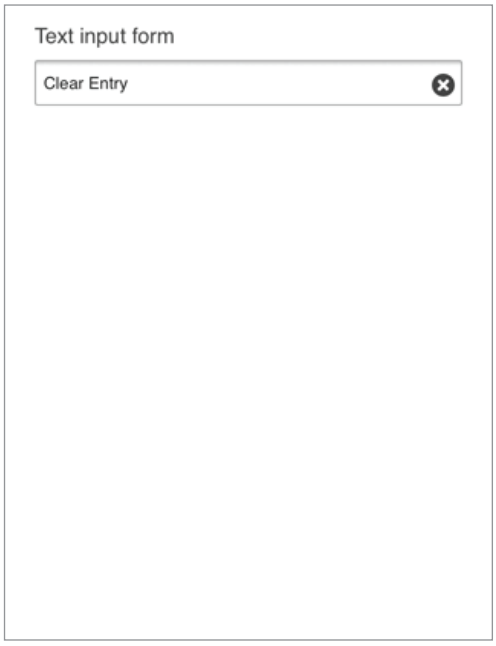


Figure 6.65 CLEAR ENTRY <patterns.jribeiro.org/patterns/clear-entry>.

Problem

Although typing on a virtual keyboard is a difficult task, resetting an input form to the default state may be no less easy. Deleting a long string of text letter by letter can be a very tedious error prone task.

Solution

Provide a button that resets the input form with one tap.

***

You should always strive to minimize users' need to input text on mobile. Like text entry, deleting long strings of text can be a very tedious task and propitious to mistakes. While, generally, operating systems have some sort of method to facilitate the clearing of input fields, such as, faster deleting on long presses, you may still provide a better and faster method for this task.

Therefore, provide a button on all free-text input fields that allows users to quickly remove previous composed text. Place that button inside the input field aligned to the right and farther enough from other targets so it is not tapped by mistake. If it is needed you can use an ICEBERG TIP to improve the efficacy of that button. A button with an "x" is succinct, unambiguous and almost a standard so it is usually favored, but if you have the space you can use a label like "Clear" or "Reset".

This pattern is based on the patterns *Text Clear Button* (Tidwell 2011) and *Clear Entry* (Hoober and Berkman 2011), so you can find additional information there.

***

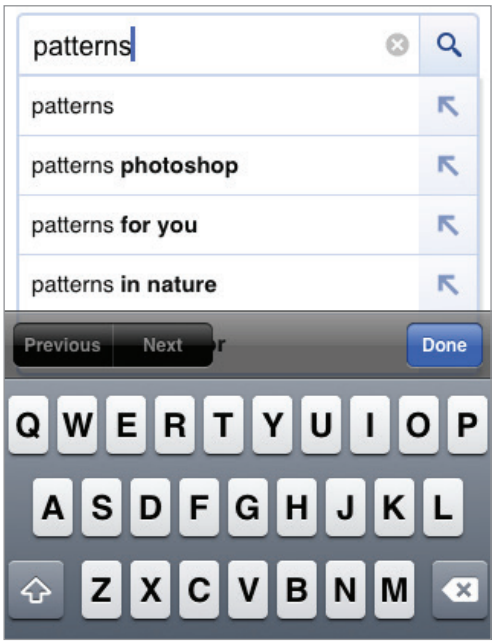


Figure 6.66 *Google* <www.google.com>, April 2012.

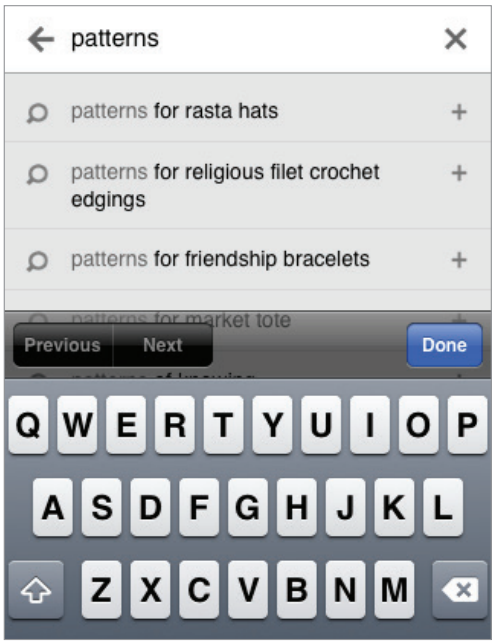


Figure 6.67 *Bing* <www.bing.com>, April 2012.

Related Patterns

- TOUCH FRIENDLY TARGET

- *Clear Entry* in *Designing Mobile Interfaces* (Hoober and Berkman 2011)
- *Text Clear* Button in *Designing Interfaces* (Tidwell 2011)

6.20 Touch Friendly Target

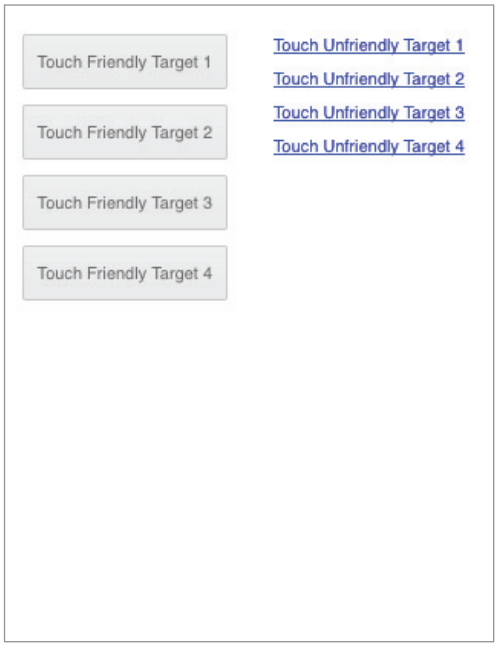


Figure 6.68 TOUCH FRIENDLY TARGET <patterns.jribeiro.org/patterns/touch-friendly-target>.

Problem

Because of the small screen, the nonexistence of tactile feedback, and the lack of precision of our *fat fingers*,[41] hitting a target on a mobile device can be a challenging task. You need to design an interface that must be effortlessly used by touch.

Solution

Design all touchable elements large enough and generously spaced so they can be easily triggered.

***

With a mouse we can easily trigger very small targets, on mobile devices, because we are using our fingers as an input device that can be a more challenging task. Our fingers are much more imprecise than a mouse, as such, you should design touchable elements large enough so users can easily interact with them.

It is frustrating when we press a button and nothing happens. Currently devices provide no haptic feedback, so users cannot know for sure if they just missed the target or there is a prob-

41 The 'fat finger syndrome' is a colloquial term used to describe the trigger of accidental actions caused by the fact that users' fingers are larger than the intended target.

lem with the website. You can reduce the problem of the lack of feedback by providing some visual response to the fact that a target was tapped; for example, changing the background color of the target.

In addition to larger targets, you need to account for the space between targets. You should have generous space between elements to minimize errors. If the implementation of this pattern leads to enormous targets, you can use an ICEBERG TIP instead.

Optimal Size

The recommended minimum size for a target differs depending on which user interface guideline we may be following. However, the optimal size should be approximately that of an adult finger, which largely have a diameter of 16mm to 20mm (Saffer 2008), but the size in pixels varies depending on pixel density:

- The iPhone Human Interface Guidelines (Apple Inc 2012), recommends a minimum of 44x44 pixels for targets. Since the release of devices with higher DPI, *Apple* updated that value to an abstract measure of 44x44 points.
- User Experience Design Guidelines for Windows Phone (Microsoft Corporation 2012) recommends a 9mm target as the ideal size for all devices across *Microsoft* platforms, and 7mm as the minimum for the height when the width of the target is larger. It also recommends 4.2mm as the minimum visual size for a touchable item; and 2mm for the space between targets.
- Nokia Developer's (Nokia Corporation 2012) resources recommends that touchable elements should be no smaller than 10x10mm. And the minimum size for target should be: 7x7mm with 1mm gaps for index finger usage; 8x8mm with 2mm gaps for thumb usage; and lists should have a minimum of 5 mm of line spacing.

***

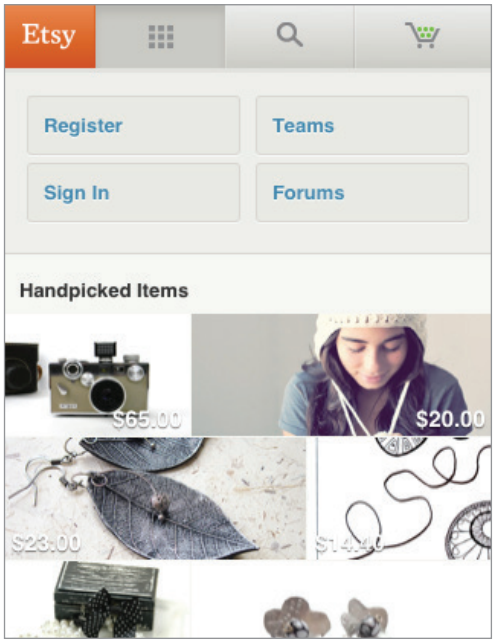


Figure 6.69 *Etsy* <www.etsy.com>, July 2012.

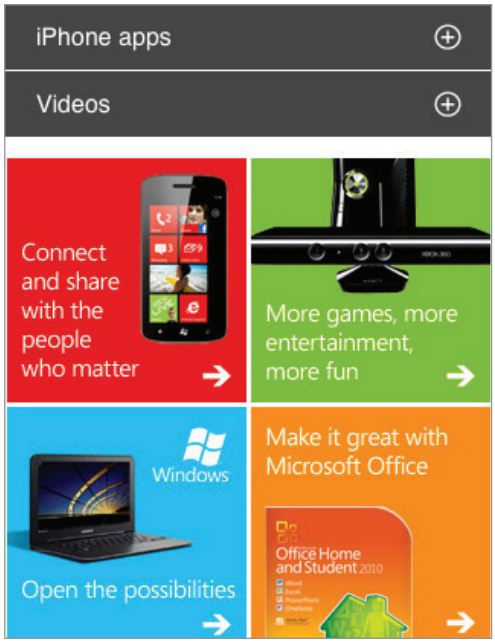


Figure 6.70 *Microsoft* <www.m.microsoft.com>, July 2012.

Related Patterns

- ICEBERG TIP
- *Generous Borders* in *Designing Interfaces* (Tidwell 2011)

6.21 Iceberg Tip

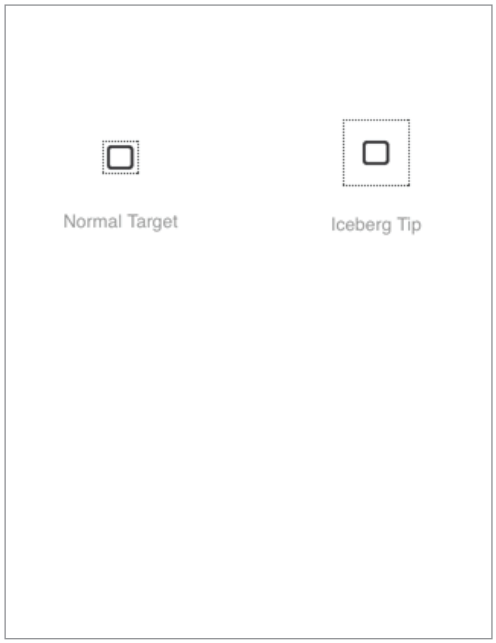


Figure 7.71 ICEBERG TIP <patterns.jribeiro.org/patterns/iceberg-tip>.

Problem

Sometimes you need to design a TOUCH FRIENDLY TARGET, but you do not want to have enormous and inelegant buttons to clutter the interface.

Solution

Design the visible part of your object with whatever size you planned, but make the real target invisible and large enough so it can be easily touched.

***

This pattern is inspired on the idea described by Dan Saffer in *Designing Gestural Interfaces* (2008), which, like the name suggests, uses the metaphor of an iceberg for describing a target that is larger than the visible area.

When designing touch friendly interfaces, touchable elements must be large enough so users can easily interact with them. However, it is not always practical, or visually pleasing to design big buttons throughout the entire interface, in fact, touch friendly buttons may sometimes look clumsy. This pattern is valuable for when we have any element, either text or image, that by itself is not large enough to be triggered without effort; or when designing big buttons may go against what was envisioned for the visible design of the interface.

The solution involves having the visible part of the element surrounded by an invisible padding area that also works as a target, i.e., only a portion of the target is visible, the rest is hidden.

One limitation of this pattern is that it cannot be used when we have several touchable elements side by side, without creating additional white space between them.

We should always try to provide the correct affordances for the interface, but because of the small scale that the visible component can reach, the simpler design, or just because users were not expecting that elements that small would work as buttons, it may happen that these elements are not be perceived as touchable. Microsoft (2012) recognizes this downside and recommends that the graphic component should be at least 4.2mm.

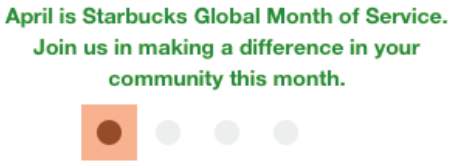


Figure 6.72 ICEBERG TIP on *Starbucks* <www.starbucks.com>, the color overlay represents the real target.

Starbucks (Figure 6.72) uses small circles with a diameter of 16 pixels as buttons for navigating through a SLIDESHOW. Those buttons do not look large enough for being easily touched; however, the real target is a 36 pixels square, which improves considerably its efficacy, though, it is still below what is usually recommended by TOUCH FRIENDLY TARGET.

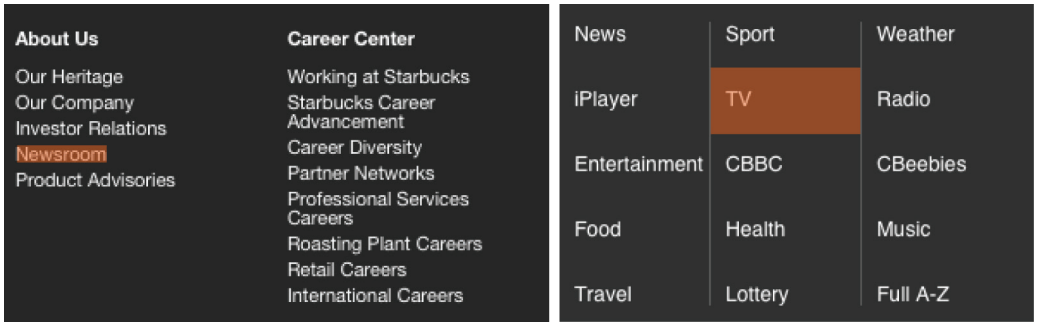


Figure 6.73 Comparison of targets on *Starbucks* <www.starbucks.com> (left) and *BBC* <www.m.bbc.co.uk/news>(right).

On the Figure 6.73, on the left image, we have a menu with links that could be improved by using this pattern. The text is too small, which by itself is not necessarily a problem, but the target occupies the same space as the text, in this case, as low as 8 pixels height; and there is not adequate space between links, which makes it almost impossible to hit the intended target on the first try. On the contrary, on the right image, we have a comparable menu, but here, while the visible part is still small, links have plenty of space between them and the real target occupies the maximum space possible.

***

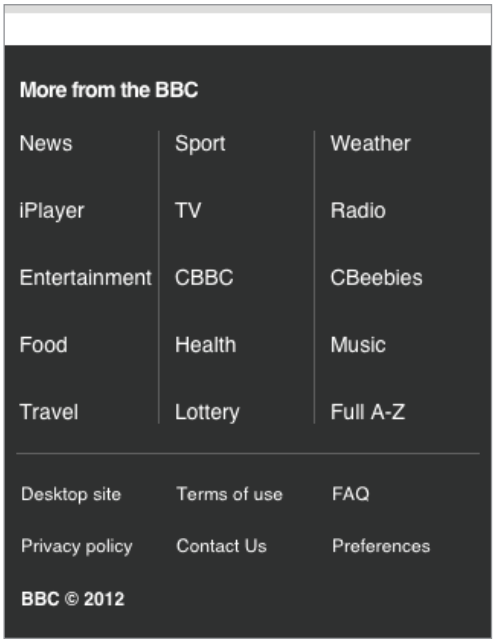


Figure 6.74 *BBC* <www.m.bbc.co.uk/news>, April 2012.

Figure 6.75 Earth Hour <www.earthhour.fr>, July 2012.

Related Patterns

- TOUCH FRIENDLY TARGET

7 CASE STUDIES

The idea of using real projects as part of the process of writing patterns was explored by some of the authors cited in this dissertation. *A Pattern Language* (Alexander 1977) is the result of the experience of working on the planning and building of the University of Oregon campus by Alexander and his team as it is documented on *The Oregon Experiment* (1975). The *Gang of Four* (1995) presents a case study of the design of a WYSIWYG editor that demonstrates how design patterns can be applied in practice. The patterns presented in *A Pattern Approach to Interaction Design* (Borchers 2001) result from the experience gained by the author during the design of user interfaces for four different projects.

In this chapter we present three case studies, where we describe the results of the experience and know-how obtained from designing websites for mobile devices. With these projects we expect to ascertain how the patterns that were previously proposed perform on real-world contexts.

The projects presented in these case studies were developed in parallel with the writing of the patterns. Neither came first than the other. The writing and design were part of an iterative process that shared information between both tasks, which was part of the whole process of capturing patterns. The design and implementation of these websites allowed us to discover new patterns, and to explore and validate the patterns that were being written, while the writing of the patterns enabled us to systematize the insights gathered throughout the design part.

We present three different projects and for each one we describe a unique pattern language obtained with the patterns in this work.

7.1 e-Learning Café

For the e-Learning Café project we had to design a website for the purpose of presenting the e-learning café spaces — leisure and study spaces open to the academic community that combines social, entertaining, study and work contexts — to the University of Porto community.

The project was developed by the IDD[42] team for whom it was proposed the redesign of the old website, which included a revised information architecture, an update to the previous layout, as well as a layout prepared to mobile phones and tablets.

Figure 7.1 Mobile and desktop versions.

42 IDD (Investigação e Desenvolvimento em Design) is a research group from the Faculty of Fine Arts of the University of Porto with whom I have been collaborating for the past two years. All projects described in this chapter were developed by this team.

The website was designed thinking on mobile devices from the beginning, which was reflected on the concerns of how the page content could reflow on different screens sizes, or how to design TOUCH FRIENDLY TARGET. We use a responsive design approach with three major breakpoints reflecting the most common screen sizes of mobile devices: less than 768 pixels for mobile devices, with some minor tweaks for the landscape view; between 768 and 1024 pixels for tablets on portrait; and more than 1024 pixels for tablets on landscape and desktop systems in general.

The information architecture of the website is rather simple. There were only four major pages: a homepage providing a general overview of the space concept, working as a hub to connect users to the other sections of the website; a dedicated page for each e-learning café that presents the information that is specific to that space; and a news and events page.

The layout was designed heavily on the vertical axis, so it was natural to design a mobile version that relied on a LINEARIZED LAYOUT. As we can see on Figure 7.1, the page was designed by stacking sections on top of each others, filling an entire horizontal stripe on the page, and distinguished by strong color contrasts. There were columns only within each section, which were linearized on the mobile view. We only used a GRID LAYOUT to list staff members (Figure 7.2); and to present the second-level menu (Figure 7.3).

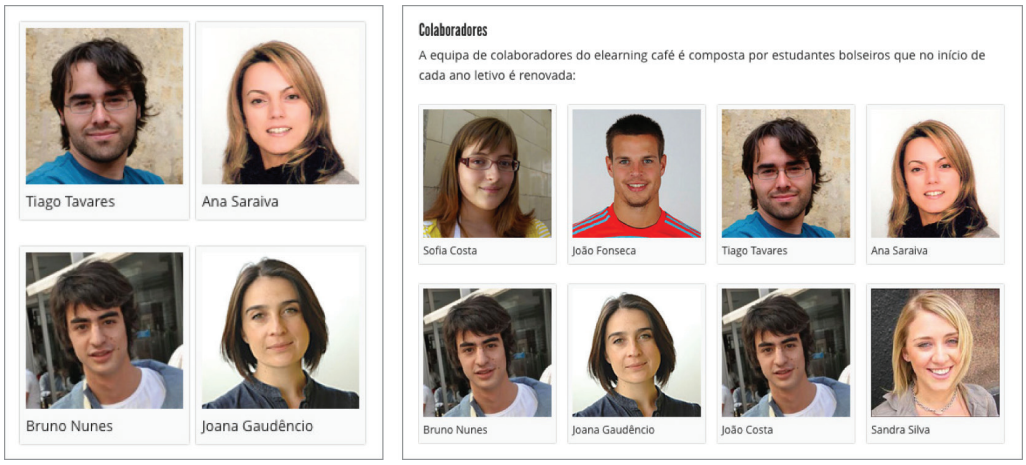


Figure 7.2 GRID LAYOUT for displaying the list of staff members (currently with placeholder content), mobile on the left, desktop on the right.

On larger screens the page header was fixed to the top of the browser window, but for mobile we decided that FIXED CONTENT would take too much space unnecessarily, so on smaller screens that element is positioned respecting the normal flow of the page.

Because the first-level menu only had four items, it was possible to present those menu items inline (Figure 7.3); however, some changes had to be made. The logo on the header that works a link to the homepage is composed by two elements: the logo mark and the typography. On smaller screens the typography is removed; only the mark is left. The names of the spaces

were shortened; only the part that clearly identifies the space stayed. Finally, the textual news and events link was changed to a calendar icon. This idea of removing or changing elements on the page while still retaining their meaning could be captured as a pattern on future work.

Figure 7.3 TOGGLE MENU for the second level menu.

The items on the second-level menu are links to subsections on the same page, which can be already accessed by simply scrolling through the page, thus, we thought that it would be more space efficient if we did not display that menu when the page loads. Nonetheless, we decided that it was useful to offer that menu, even if it were just as a page index, so, it is still possible to access it through a TOGGLE MENU (Figure 7.3) by tapping on the active item of the first-level menu.

The news section had a rather different layout. On the desktop view this page was composed by a column layout where blocks with the latest news were floated to create a more dynamic composition; on the mobile view the layout was linearized through an INFINITE LIST (Figure 7.4).

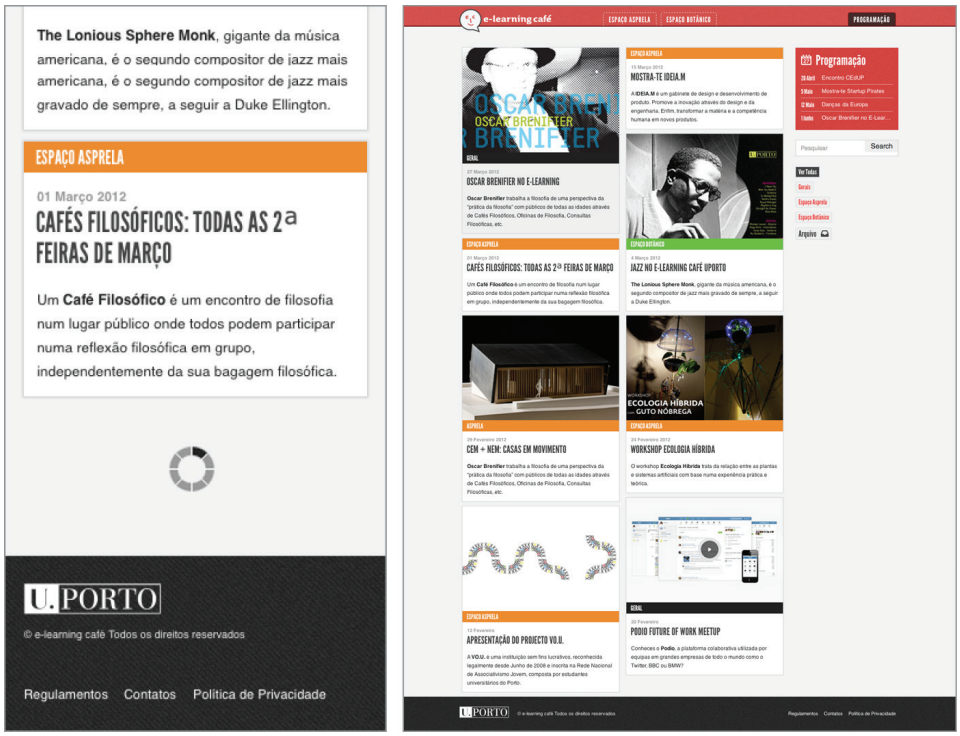


Figure 7.4 INFINITE LIST, mobile on the left, desktop on the right.

7.2 Moodle 2

This project consisted on the design of a new theme for the version 2 of Moodle — an open source learning management system —, and was developed by the IDD team in collaboration with the University of Porto (UP). The main focus of this project has been on the desktop version, for which we designed a new layout to meet the requirements of the latest version of Moodle. It was decided that to take advantage of the new version as an opportunity to develop an accompanying mobile version. The mobile version had to incorporate the visual design of the desktop version to offer a unified image across platforms. Because Moodle has an enormous amount of functionalities, many of which are too complex to implement on a mobile device, and because of time constraints, on a first stage we decide to concentrate on a smaller set of functionalities. Based on the results of a survey done to current Moodle users from the UP, in which it was asked which were the features that they wish, or expect to have on a mobile version, we started the development of the mobile version with the following functionalities:

- consult information left by the teacher;
- post or reply on forums;
- consult the latest messages and send new ones.

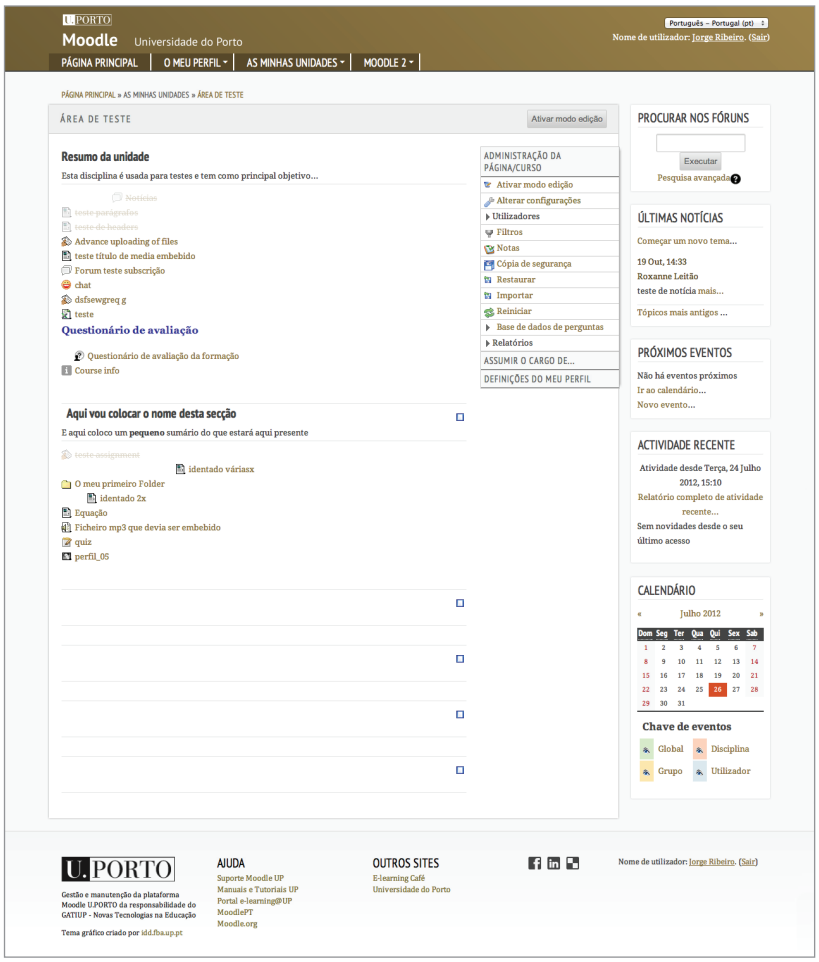


Figure 7.5 Desktop version of Moodle 2 theme for the University of Porto.

More complex features such as managing the course (e.g., editing course information, adding projects, grading students) were omitted. These tasks are already complex on the desktop layouts so we opted to remove it from the mobile version. The mobile version was thought mainly as a platform to access information on the go, and to perform simple tasks.

Unlike the other projects mentioned on this chapter that rely on a responsive design approach, the mobile view for this website has a custom layout. Moodle detects the browser user agent and serves a dedicated theme to mobile devices. However, because of some platform limitations, while the visual design between the mobile and desktop version are considerably different, the content and markup is almost the same. This is a problem because the mobile theme receives content that is unnecessary, or that it not formatted as it should. We can recognize that for complex layouts it may be more convenient to have a dedicated mobile version, which can have distinct content, with proper, and optimized markup and assets. That supports what, for example, Nielsen (2012b) has been recommending: to build a separated mobile-optimized site.

Figure 7.6 VERTICAL LIST.

Figure 7.7 THUMBNAIL LIST.

The use of lists to present information has been a recurring practice on the design of mobile web sites. For example, we used a VERTICAL LIST (Figure 7.6) to present the available courses, and a THUMBNAIL LIST (Figure 7.7) to list the course contents, or and EXPANDING LIST (Figure 7.8) for presenting complementary information.

The desktop version has a sidebar with complementary information formed by thematic blocks that can be arranged and managed by the teacher, thus being different for every course. Since they do not present critical information we opted to relegate it to the end of the page, where it is presented as an EXPANDING LIST (Figure 7.8), in which the block title is used as a toggle to display the content of that block.

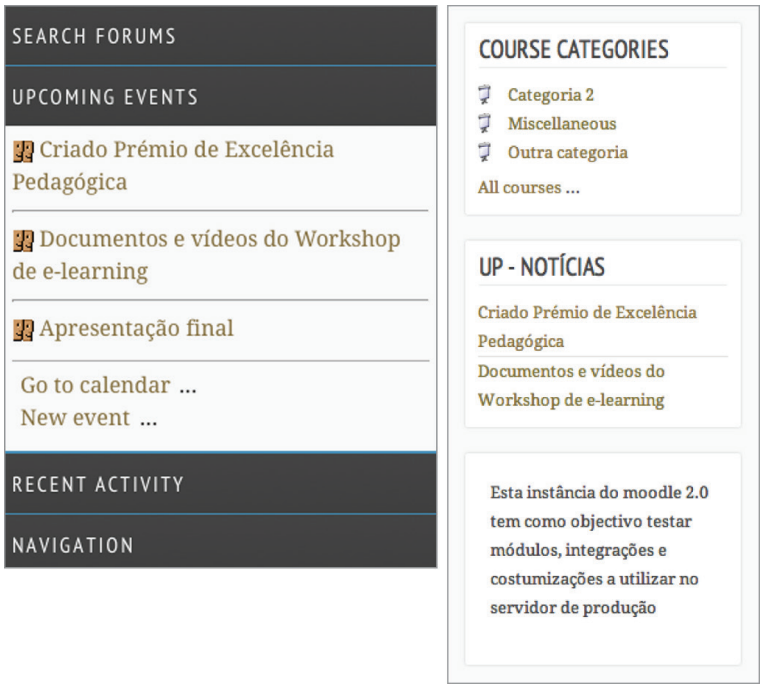


Figure 7.8. EXPANDING LIST, mobile on the left, dektop on the right.

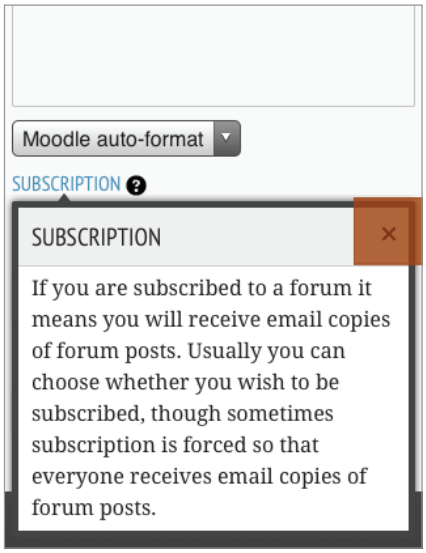


Figure 7.9. ICEBERG TIP for closing a DROPDOWN.

We used an ICEBERG TIP to improve the efficacy of tappable links on different design elements. For example, on Figure 7.9 the visible target for closing a DROPDOWN is 10 pixels square while the real target is a 50 pixels square, as can be seen with the color overlay.

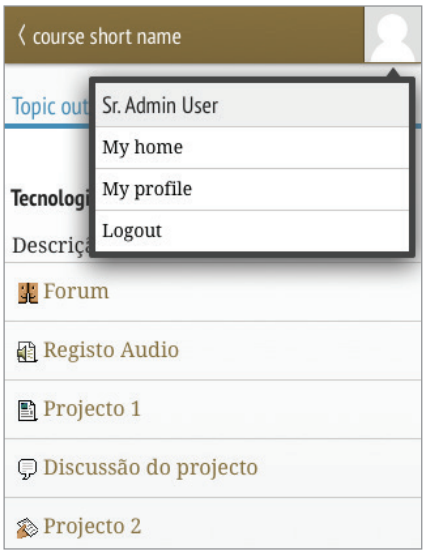


Figure 7.10 DROPDOWN.

The DROPDOWN was useful to provide access to additional information without to clutter the interface with additional content. For example, on Figure 7.9 a DROPDOWN was used to display contextual help; on Figure 7.10 it was used to provide access to additional functionalities.

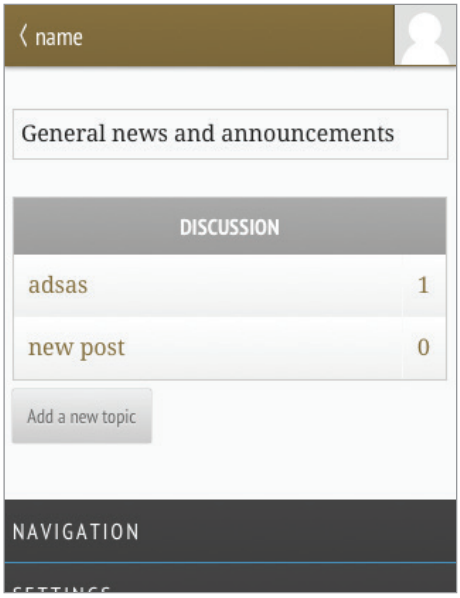


Figure 7.11 ABRIDGED TABLE.

Moodle recurs occasionally to tables for presenting information, but most of those tables are very extensive so do not work properly on smaller screens. To overcome this problem we had to simplify them, which in practice meant to remove columns that we considered less important. This was the elemental idea for what later became the ABRIDGED TABLE. For example, on Figure 7.11 we have a table with two columns that lists forum posts. The original table was

longer but was simplified by omitting two additional columns, so it fitted comfortably on the mobile view. Although the ABRIDGED TABLE recommends the use of a button to toggle the visibility of the hidden columns, we opted to not provide it, because the information contained in those columns was not essential to the comprehension and use of the table.

7.3 University of Porto

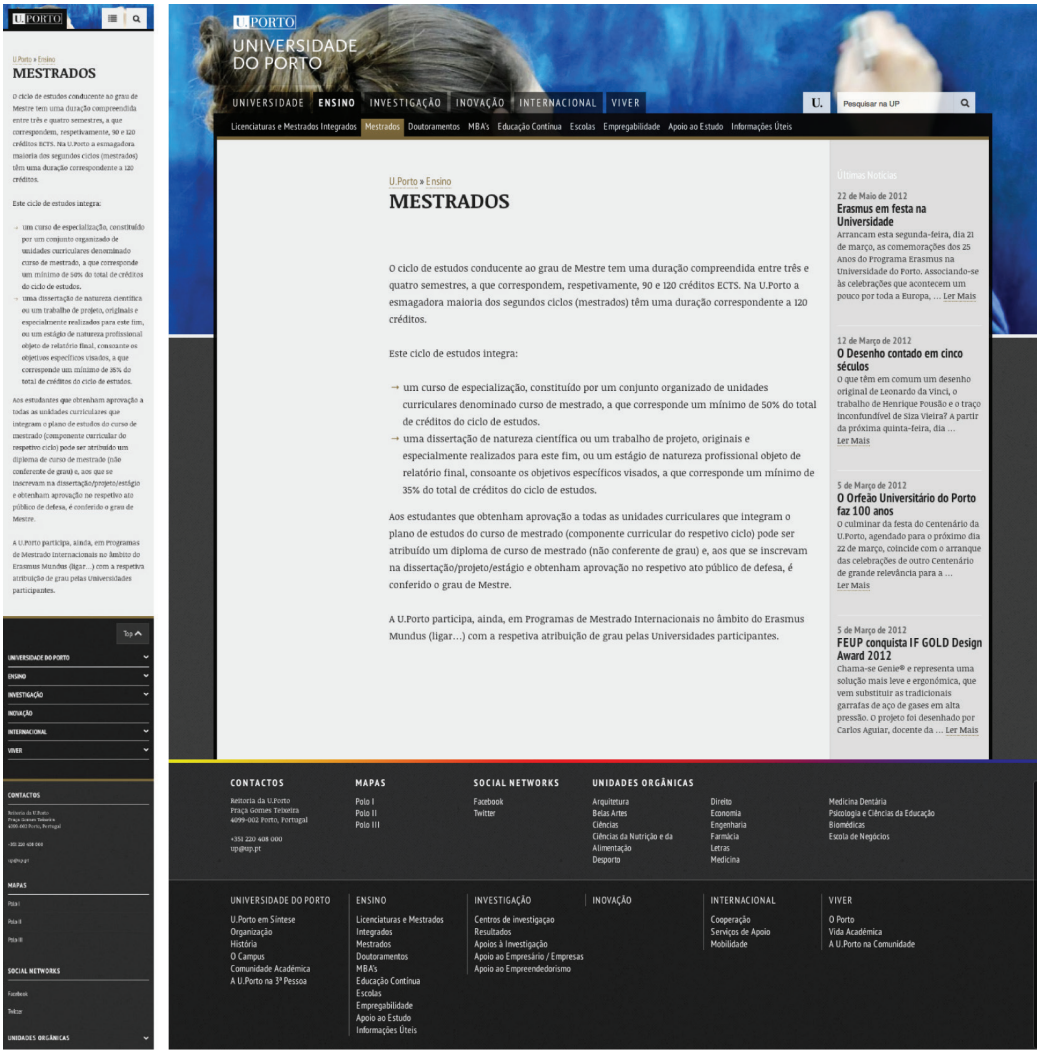


Figure 7.12 Desktop versus mobile version.

The last project focused on the design of the mobile layout to the new website of the University of Porto (UP). For this project the IDD team worked in collaboration with elements from UP that were responsible by the technical implementation of the website, as well as content producers. Our team was responsible by initial research, focused on a user-centered design, in which we developed a content analysis of the previous website, interviews with project stakeholders,

an inquiry to the UP community about the old site, development of usability, accessibility and mobile guidelines, creation of personas, and the proposal of a new information architecture. The second stage consisted on the development of wireframes, visual designs with different levels of fidelity, and the development of a functional prototype; all this work was assisted by usability testing. At the time of the writing project is still ongoing, so the screenshots and patterns presented here may not be part of the final version.

The project was focused on the design of the desktop layout, but from the beginning all the design decisions took into consideration the mobile version. For this project, because the content of the mobile version would not differ drastically from the desktop version we opted to design the mobile view with a responsive design approach.

Unlike, for example, the e-Learning Café project, this website had a quite complex information architecture, with a navigation that had three hierarchic levels. The navigation was indeed the most difficult and changeling part to adapt successfully to the small screen, since the rest of the layout was implemented straightforwardly with a LINEARIZED LAYOUT (Figure 7.12). The mobile layout was simplified through the removal of complementary information such as the sidebar dedicated to the news. The website header was also simplified on the mobile version: only three items, which were designed as a TOUCH FRIENDLY TARGET, were kept. The branding image was changed to simpler and smaller version; the menu was removed from the top of the page; and the search form was changed. Similar solution to that already used on the e-Learning Café project.

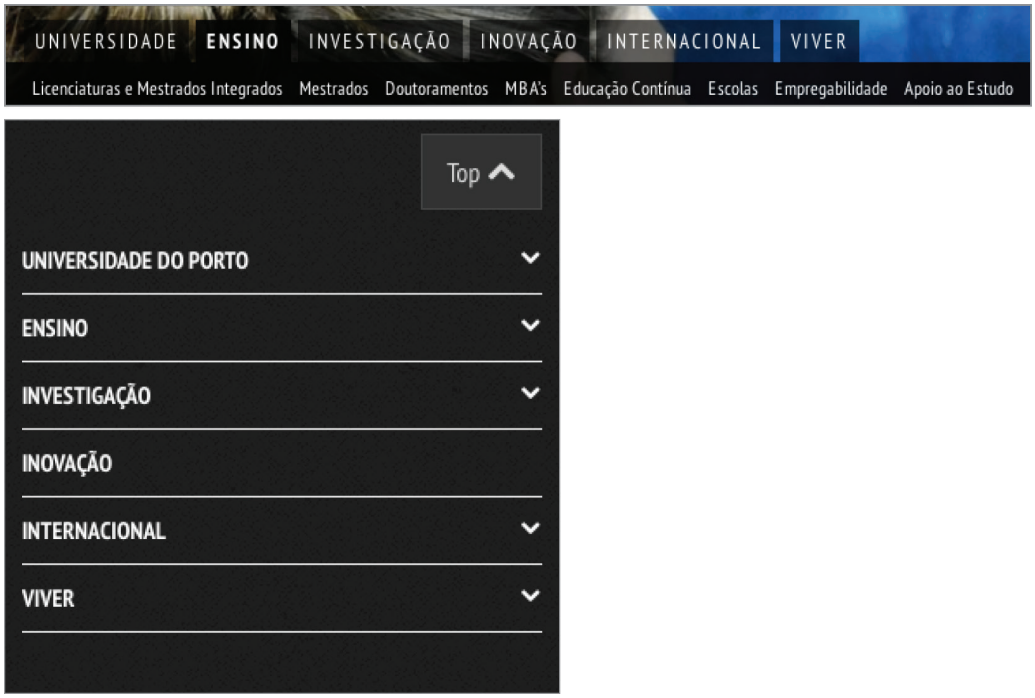


Figure 7.13 Desktop versus mobile navigation.

On the desktop version we had a large footer that was used to present a site-wide navigation of the top two levels; and to present additional links to important sections of the site. This complex footer became particularly useful for the implementation of the navigation on mobile. Since we already had the entire navigation replicated at the bottom of the page, and because the menu was quite extensive, we decided to use a JUMP MENU (Figure 7.13). The implementation was fairly simple: we only had to hide the menus on the top of the page and to provide a link to the bottom (Figure 7.15).

Figure 7.14 Second-level menu expanded on the left, and third-level menu expanded on the right.

Although already complex on a wide screen, efficiently present a three-level navigation on a small screen is even more difficult. To achieve that task we extended the JUMP MEMU with an EXPANDING LIST (Figure 7.14). When the page loads only the first level is visible, so that the page does not extends excessively. However, users can expand the menu by tapping the downward arrow — which changes to an upward arrow when the item is expanded — at the right of the list item (Figure 7.14), and thus access the additional navigation levels. Ideally, the entire row would work as a toggle for the EXPANDING LIST, however, those headings were also links, so we had to have in the same row a link and a toggle button. Because the list item text and toggle icon were both too small, we used an ICEBERG TIP to improve the efficacy of their tappable areas; the real target can be seen on Figure 7.14 through the color overlay.

Figure 7.15 DROPDOWN used to present the search input, closed on the left, opened on the right.

To provide easy access to the website search, we decided to display the search form on top of the page, but because we did not wanted to highlight it too much, we used a DROPDOWN (Figure 7.15) that was only unveiled when the search icon was tapped. To facilitate the removal of text entries, the search input also had a CLEAR ENTRY button (Figure 7.15), which was also improved by the use of an ICEBERG TIP (Figure 7.15, right image).

8 CONCLUSIONS

The purpose of this work was to research on the design of mobile websites in order to propose new tools to help designers, as well as non-designers developing for mobile devices. We approached this problem through a pattern-based methodology, which resulted on the design patterns that were here presented.

We research on the current state of mobile devices regarding the domain of the web, in which we explored the limitations and possibilities that these devices impose to the design of sites. We consider to had provided a comprehensive overview of the current state of the mobile web, however, the topics addressed in this research regard subjects in a field that has been evolving at a fast pace in the last few years, hence these information needs to be revised once new devices and technologies emerge. Nevertheless, we believe that some of the points made will not change significantly, for instance: limitations concerning the smaller size of the screen are inherently part of the characteristics that define a mobile device; as well as those related to human factors, such as target sizes.

We conducted an extensive comparative analysis of different patterns libraries, from where we can apprehend how different authors have been approaching the same problem; from analysis also we can also extract a historical overview of how patterns, and pattern libraries evolved over the years.

Based on the initial research, we proposed a new template for the organization of patterns, which was then used to format the patterns of this work. Although the template does not differ substantially from the ones used by other authors, there are some differences that we consider that contribute to a more clear and precise understanding of interaction design patterns, namely, the illustration that complements each pattern. We developed, in a systematic manner, a series of interactive illustrations that we regard as important elements to a successful representation of patterns, which can be useful, as well, to other interaction design pattern libraries.

With the purpose of offering a more accessible means for viewing the patterns, but mostly with the necessity of presenting the interactive illustrations, we developed a website from where patterns can be consulted — www.patterns.jribeiro.org. Currently the website is, above all, a practical tool for presenting the interactive illustrations; however it might be worthwhile

to continue its development by providing additional context about this work, and by centering the improvement and expansion of the pattern library around it. The website can become a more valuable tool as it reaches a broader audience.

The research presented in this work culminated on the twenty-one design patterns that were here proposed. We expect to have been able to capture a comprehensive sample of patterns, large enough to help one get through an entire design project. However, this work is far from finished, these patterns need to be viewed as a work in progress, in a way that the current ones can be expanded and new ones can be added.

Through the course of this project, we were developing, simultaneously, three different projects that allowed us to consolidate and validate the development of patterns. These works also offered us some insights for new patterns that may be recorded on future work. For example, as it was previously described, in a few occasions we solved the problem of limited space by changing elements (or removing part of it) with ones that were more space efficient, while retaining the original meaning of that element. Although it did not have the time to make to the final list of patterns, this idea was documented for future work.

Working with these patterns in real design projects allowed us to realize that some patterns were more recurrent than others, for which we consider to have found a respectable level of invariance. For instance, it is almost guaranteed that every design will have a VERTICAL LIST, or any of its variations. Likewise, the TOUCH FRIENDLY TARGET and the ICEBERG TIP, because constrained by human factors will expectably remain relevant to any design. Nonetheless, all patterns that are here presented comprise some level of confidence, in the way that, somehow, we consider that each one, on its own manner, is a possible solution for a concrete design problem. Evidently, we cannot assert the same level of confidence to all of them. Therefore, if the development of the pattern library continues, while not mandatory, it would be important and advisable, to explore the conception of some sort of rating system that could offer a more systematic classification for each pattern.

We also would like to mention that the resulting patterns were adapted to a paper format and submitted to PLOP[43] 2012. Our submission was accepted, being now in the shepherding process — a stage in which the paper is reviewed by an experienced pattern author in order to improve the final version. The patterns that were presented already include some of the insights suggested by our shepherdess Dr. Alejandra Garrido.

We developed this initial set of pattern with the hope to contribute to the design of better and more satisfying mobile experiences, but by their nature, patterns and pattern languages are unfinished artifacts, hence this work is just the foundation for an iterative, and ongoing process of development of tools that are aware of how the web evolves.

---

43 Pattern Languages of Programs (PLOP), an annual conference on design patterns.

BIBLIOGRAPHY

Alexander, Christopher. *Notes on the Synthesis of Form*. Cambridge, Massachusetts: Harvard University Press, 1964.

———. *The Oregon Experiment*. New York: Oxford University Press, 1975.

———. *The Timeless Way of Building*. New York: Oxford University Press, 1979.

Alexander, Christopher, et al. *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford University Press, 1977.

Apple Inc. "Platform Characteristics." iOS Human Interface Guidelines, 2012. 3 June 2012. <http://developer.apple.com/library/ios/#DOCUMENTATION/UserExperience/Conceptual/MobileHIG/Characteristics/Characteristics.html>.

Behrens, Christian. "The Form of Facts and Figures." Potsdam University of Applied Sciences, 2008a.

———. "Info Design Patterns." 2008b. 18 January 2012. <http://infodesignpatterns.com>.

Borchers, Jan O. *A Pattern Approach to Interaction Design*. Chichester: Wiley Publishing, 2001.

———. "Interaction Design Patterns: Twelve Theses." *Position Paper, Workshop "Pattern Languages for Interaction Design: Building Momentum"*, CHI 2000. 2000.

Budiu, Raluca, and Jakob Nielsen. "Usability of iPad Apps and Websites." 2010.

Carvalhais, Miguel. "Learning and Studying Interaction Design through Design Patterns." *15th Conference on Pattern Languages of Programs*. 2008.

Cisco Systems. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2010–2015." 2011. 3 June 2012. <http://newsroom.cisco.com/dlls/ekits/Cisco_VNI_Global_Mobile_Data_Traffic_Forecast_2010_2015.pdf>.

Clark, Josh. *Tapworthy: Designing Great Iphone Apps*. Sebastopol, California: O'Reilly Media, 2010.

Clarke, Andy. "We Need a Standard Show Navigation Icon for Responsive Web Design." 2012. 3 June 2012. <http://stuffandnonsense.co.uk/blog/about/we_need_a_standard_show_navigation_icon_for_responsive_web_design>.

Cooper, Alan, Robert Reimann, and David Cronin. *About Face 3: The Essentials of Interaction Design*. Indianapolis, Indiana: Wiley Publishing, 2007.

Coyier, Chris. "Responsive Data Tables." CSS Tricks, 2011. 10 July 2012. <http://css-tricks.com/responsive-data-tables>.

Crumlish, Christian, and Erin Malone. *Designing Social Interfaces: Principles, Patterns, and Practices for Improving the User Experience*. Sebastopol, California: O'Reilly Media, 2009.

Firtman, Maximiliano. *Programming the Mobile Web*. Sebastopol, California: O'Reilly, 2010.

Fling, Brian. *Mobile Design and Development: Practical Concepts and Techniques for Creating Mobile Sites and Web Apps*. Sebastopol, California: O'Reilly Media, 2009.

Frost, Brad. "Responsive Navigation Patterns." Brad Frost Web, 2012. 3 June 2012. <http://bradfrostweb.com/blog/web/responsive-nav-patterns>.

Gamma, Erich, et al. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison-Wesley Professional, 1995.

Gartner Inc. "Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent." 2010. 3 June 2012. <http://gartner.com/it/page.jsp?id=1466313>.

Google Inc. "Platform Versions." 2012. 3 June 2012. <http://developer.android.com/resources/dashboard/platform-versions.html>.

Hoober, Steven, and Eric Berkman. *Designing Mobile Interfaces*. Sebastopol, California: O'Reilly Media, 2011.

Instituto Superior Técnico. "Banco de Padrões de Design." 2010. 4 January 2012. <http://bpd.ist.utl.pt>.

Koch, Peter-Paul. "Smartphone Browser Landscape." A List Apart, 2010. 4 June 2012. <http://alistapart.com/articles/smartphone-browser-landscape>.

Marcotte, Ethan. Responsive Web Design. New York: A Book Apart, 2011.

———. "Responsive Web Design." A List Apart, 2010. 11 July 2012. <http://alistapart.com/articles/responsive-web-design>.

Microsoft Corporation. "Interactions and Usability with Windows Phone." User Experience Design Guidelines for Windows Phone, 2012. 3 June 2012. <http://msdn.microsoft.com/en-us/library/hh202889(v=VS.92).aspx>.

Neil, Theresa. Mobile Design Pattern Gallery. Sebastopol, California: O'Riley Media, 2012.

Nielsen, Jakob. "Browser and Gui Chrome." Use it, 2012a. 11 June 2012. <http://useit.com/alertbox/ui-chrome.html>.

———. "Mobile Site Vs. Full Site." 2012b. Use it. 17 June 2012. <http://useit.com/alertbox/mobile-vs-full-sites.html>.

———. "Mobile Usability Update." 2011. Use It. 4 June 2012. <http://useit.com/alertbox/mobile-usability.html>.

———. "Tabs, Used Right." 2007. Use It. 3 June 2012. <http://useit.com/alertbox/tabs.html>.

Nielsen, Jakob, and Donald A. Norman. "Gestural Interfaces: A Step Backwards in Usability." 2010. *Interactions*. 5 17. <http://jnd.org/dn.mss/gestural_interfaces_a_step_backwards_in_usability_6.html>.

Nokia Corporation. "Scale and Positioning of Controls." Nokia Developer, 2012. 3 June 2012. <www.library.developer.nokia.com/index.jsp?topic=/S60_5th_Edition_Cpp_Developers_Library/GUID-5486EFD3-4660-4C19-A007-286DE48F6EEF.html>.

Norman, Donald A. "Natural User Interfaces Are Not Natural." 2010. 3 June 2012. <http://www.jnd.org/dn.mss/natural_user_interfa.html>.

OpenSignalMaps. "Android Fragmentation Visualized." 2012. 3 June 2012. <http://opensignalmaps.com/reports/fragmentation.php>.

Pattern Factory. "Patternry Open Library." 2009. 18 January 2012. <http://patternry.com>.

Saffer, Dan. *Designing Gestural Interfaces: Touchscreens and Interactive Devices*. Sebastopol, California: O'Reilly Media, 2008.

Schümmer, Till, and Stephan Lukosch. *Patterns for Computer-Mediated Interaction*. Chichester, England: Wiley Publishing, 2007.

Scott, Bill, and Theresa Neil. *Designing Web Interfaces: Principles and Patterns for Rich Interactions*. Sebastopol, California: O'Reilly Media, 2009.

Smith, David. "OS 5.1.1 Upgrade Stats." 2012. 3 june 2012. <http://david-smith.org/blog/2012/05/11/ios-5-dot-1-1-upgrade-stats>.

StatCounter. "Statcounter Global Stats." 2012. 26 July 2012. <http://statcounter.com>.

Tidwell, Jenifer. "Common Ground: A Pattern Language for Human-Computer Interface Design." 1999. <http://mit.edu/~jtidwell/common_ground.html>.

———. *Designing Interfaces: Patterns for Effective Interaction Design*. Sebastopol, California: O'Reilly Media, 2005.

———. *Designing Interfaces: Patterns for Effective Interaction Design*. 2005. 2nd ed. Sebastopol, California: O'Reilly Media, 2011.

Toxboe, Anders. "UI-Patterns." 2007. 18 January 2012. <http://ui-patterns.com>.

van Duyne, Douglas K., James A. Landay, and Jason I. Hong. *The Design of Sites: Patterns for Creating Winning Websites*. 2002. 2nd ed. Upper Saddle River, New Jersey: Prentice Hall, 2006.

W3Schools. "Browser Statistics." 2012. 3 June 2012. <http://w3schools.com/browsers/browsers_stats.asp>.

Wroblewski, Luke. *Mobile First*. New York: A Book Apart, 2011.

Yahoo! "Yahoo! Design Pattern Library." 2006. 18 January 2012. <http://developer.yahoo.com/ypatterns>.